Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Schwarzmann

Sinteza digitalnih mikrofluidnih biočipov

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Jurij Mihelič

Ljubljana 2014

Rezultati diplomskega dela so intelektualna lastnina avtorja. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Digitalni mikrofluidni biočipi se uporabljajo kot alternativa klasičnim medicinskim preiskavam. Njihova glavna prednost je predvsem majhnost v primerjavi z laboratorijskimi napravami, kar posledično omogoča prenosljivost in zmožnost analize zelo majhne količine snovi.

V diplomski nalogi opišite področje digitalne mikrofluidike in preučite zgradbo mikrofluidnih biočipov. Osredotočite se na digitalne mikrofluidne biočipe in na njihovo sintezo. Predstavite pomembnješe vrste problemov, ki se porajajo pri njej, in algoritme za njihovo reševanje. Poleg tega opišite tudi simulator sinteze in njegovo uporabo ponazorite s primerom.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anže Schwarzmann, z vpisno številko **63110313**, sem avtor diplomskega dela z naslovom:

Sinteza digitalnih mikrofluidnih biočipov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Jurija Miheliča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14. september 2014 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LOC	laboratory-on-a-chip	laboratorij na čipu
MEMS	microelectromechanical system	mikroelektromehanski sistem
DMFB	digital microfluidics biochip	digitalni mikrofluidni biočip
DEP	dielectrophoresis	dielektroforeza
EWOD	electrowetting-on-dielectric	elektroomočljivost z dielektričnostjo
IPP	independent-path priority	prioriteta neodvisne poti
CPP	critical-path priority	prioriteta kritične poti
SSS	static synthesis simulator	statična simulacija sinteze

Povzetek

V diplomskem delu je predstavljen digitalni mikrofluidni biočip, ki je namenjen izvajanju preizkusov različnih človeških in okoljskih tekočin. Predhodnik digitalnega mikrofluidnega biočipa je zvezno pretočni mikrofluidni biočip, ki temelji na mehanskih komponentah za premikanje tekočin. Osnova za razvoj mikrofluidnih biočipov sta mikrofluidika in laboratorij na čipu, ki omogočata majhno velikost biočipov in prenosljivost naprav. V diplomskem delu so predstavljene zaporedne stopnje sinteze, ki ponazarjajo delovanje digitalnega mikrofluidnega biočipa. Splošni predstavitvi sinteze sledi opis vsake stopnje in algoritma posebej. Na koncu diplomske naloge je predstavljen simulator, s katerim lahko izvajamo sintezo in prikazujemo njene rezultate.

Ključne besede: digitalni, mikrofluidika, biočip, sinteza.

Abstract

This bachelor's thesis presents a digital microfluidic biochip that is intended for carrying out tests on various human and environmental fluids. The predecessor of the digital microfluidic biochip is the continuous-flow microfluidic biochip, which is based on mechanical components for fluid movement. The basis for developing microfluidic biochips is microfluidics and the lab-on-a-chip, which make possible small biochips and device portability. This thesis presents sequential steps of synthesis that illustrate the operation of a digital microfluidic biochip. The general presentation of the synthesis is followed by a separate description of each step and algorithm. The thesis concludes by presenting a simulator that can be used to carry out synthesis and present its results.

Keywords: digital, microfluidics, biochip, synthesis.

Poglavje 1

Uvod

Razvoj čipov je za človeštvo zelo pomemben, saj omogoča razvijanje različnih sistemov, ki pripomorejo k lažjemu življenju. Z razvojem mikrotehnologije se je velikost čipov zmanjšala, kar je pripomoglo k večji prenosljivosti naprav.

Digitalne mikrofluidne biočipe so začeli razvijati zaradi velikih stroškov pri diagnozah za različne bolezni in okvare, kot so srčno-žilne težave, rak, diabetes in virus človeške imunske pomanjkljivosti. Za delovanje digitalnih mikrofluidnih biočipov potrebujemo različne algoritme, od katerih je odvisen čas izvajanja preizkusa. Pomembne so tudi različne optimizacije za vsak algoritem in povezovanje algoritmov med seboj v skupno celoto.

V diplomskem delu bomo uporabljali besedo *preizkus* (angl. assay) za celoten postopek izvajanja na digitalnem mikrofluidnem biočipu. Preizkus izvira predvsem iz kemijskega in fizikalnega področja, kjer se uporablja za postopek, s katerim se kaj ugotovi.

V nadaljevanju tega poglavja sledi splošna predstavitev mikrofluidike in laboratorijev na čipu, ki sta osnovi za razvoj mikrofluidnih biočipov. V poglavju 2 sta predstavljeni obe različici mikrofluidnega biočipa: zvezno pretočni in digitalni. Zvezno pretočni mikrofluidni biočip je sestavljen iz mehanskih komponent, ki skrbijo za delovanje. Ob razvoju tehnologij je zvezno pretočni mikrofluidni biočip zamenjal digitalni mikrofluidni biočip. Digitalni temelji na digitalnem vezju, ki za delovanje potrebuje programsko opremo. V diplomskem delu je ločeno opisan strojni in programski nivo, ki sta odvisna eden od drugega. V poglavju 3 sledi glavni del diplomskega dela, kjer je opisana sinteza digitalnega mikrofluidnega

biočipa. Sintezo sestavljajo algoritmi za razvrščanje operacij, postavljanje modulov in usmerjanje kapljic. Te tri metode so najprej na splošno opisane, kasneje pa je predstavljen in opisan en algoritem za vsako metodo. V zadnjem, to je v razdelku 3.5, je predstavljen simulator, ki simulira izvajanje preizkusa na digitalnem mikrofluidnem biočipu in en preprost primer sinteze preizkusa.

1.1 Mikrofluidika

Mikrofluidi so različne tekočine v majhnih količinah. Te se merijo v mikro, nano, piko in femto litrih[1]. Mikro pomeni tudi majhno velikost prenosne poti in s tem tudi čipa ter majhno porabo energije. Oboje omogoča, da postane naprava za izvajanje različnih preizkusov prenosljiva. *Mikrofluidiko* lahko ločimo na znanost, ki se ukvarja z natančno kontrolo, vodenjem in manipulacijo tekočin na majhnih površinah, in tehnologijo izdelovanja mikrofluidnih naprav za preizkuse. Razvijati se je začela iz mikroelektromehanskih sistemov okoli leta 1980 [16]. Prvi pomemben proizvod, ki temelji na mikrofluidiki, je bila brizgalna glava tiskalnikov. Današnje prioritetno področje mikrofluidike je medicina, predvsem za različne analize človeških tekočin. Uporablja se za razvoj:

- mikropogonov – mehanske naprave, kot so mikročrpalke, ki povzročajo silo za premikanje tekočin, in mikroventili, ki skrbijo za usmerjanje tekočin,
- mehanike – različne mikronaprave, kot so mikromešalniki, mikrodelilniki,
- brizgalnih glav za tiskalnike,
- čipov,
- laboratorijev na čipu,
- mikrotermalne tehnologije – različne mikronaprave, kot so detektorji, grelniki.

Pri tem sodelujejo različna področja:

- računalništvo – razvija aplikacije za nadzorovanje mikrofluidov,

[1]Pomen fizikalnih količin: $1l = 10^6\mu l = 10^9 nl = 10^{12} pl = 10^{15} fl$.

- inženirstvo – raziskuje in razvija način izdelave mikrokanalov in celotnih čipov,
- fizika – raziskuje fizikalne osnove toka tekočine skozi mikrokanale,
- kemija – raziskuje različne snovi za izdelavo čipov in različnih preizkusov,
- nanotehnologija – sodeluje s primerjavami med nanotehnologijo in mikrotehnologijo,
- biotehnologija – sodeluje pri zniževanju stroškov izdelave in testiranj.

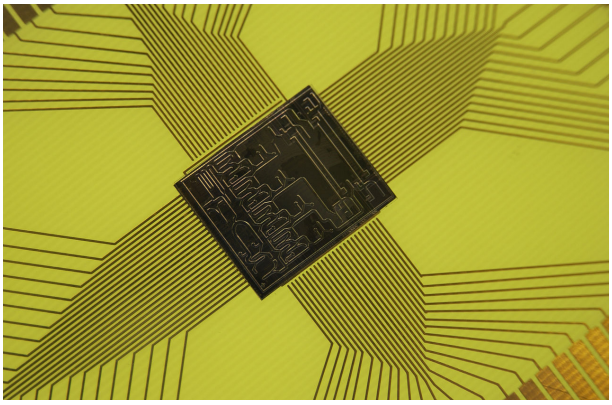
Pojavljajo se razlike med mikrofluidi in makrofluidi, predvsem v dejavnikih, kot so gravitacija, površinska napetost in izgubljanje energije. Posledica tega je lahko napačno mešanje tekočin, ki pripelje do napačnih končnih rezultatov.

1.2 Laboratorij na čipu

Laboratorije na čipu (angl. laboratory on a chip – LOC) so začeli razvijati po odkritju mikrotehnologije, ki je omogočala velikost čipov samo nekaj kvadratnih centimetrov [6]. Mikrotehnologija omogoča, da je na enem LOC-u integriran eden ali več procesov, ki jih je bilo sicer možno izvajati samo z veliko laboratorijsko opremo.

LOC je sestavljen iz dveh delov. Prvi del je čip, izdelan v večini iz stekla, ki vsebuje pasivne fluidne funkcije, kot so oskrba z *vzorcem* in *reagentom*, ter nadzor napetosti in optično odkrivanje. Reagent je snov, ki ob dodajanju k vzorcu, med izvajanjem preizkusa, spremeni lasnosti vzorca. S spremembo vzorca dokazujemo določene snovi v vzorcu. Drugi del je prava površina laboratorijskega čipa z glavnimi funkcijami, združenimi v sistem za nadzor čipa.

Zaradi majhne velikosti potrebujemo za preizkus tudi majhno količino reagenta in vzorca oziroma lahko z majhno količino vzorca naredimo več različnih preizkusov. Z manjšimi količinami vzorca oziroma reagenta se poveča pretočnost snovi, kar omogoča krajši čas delovanja LOC-a. Poveča se tudi občutljivost na majhne spremembe v temperaturi, kar ni zaželjeno. V primeru, da temeperatura vzorca ali reagenta ni taka, kot je bila predpisana v preizkusu, rezultati preizkusa ne bodo



Slika 1.1: Laboratorij na čipu Vir: [3].

točni. Nadzorovanje snovi je lažje v manjših količinah in potrebne je manj energije za premikanje snovi po kanalih v LOC-u.

Poveča se tudi varnost ljudi, ki delajo z napravo, saj delajo z majhnimi količinami snovi, ki so lahko nestabilne, radioaktivne ali povzročajo eksotermne[2] reakcije. Omejen je tudi prostor, kjer se snov lahko premika, saj so naprave majhne in s tem privarčujemo na prostoru. Pomembna je tudi učinkovitost, saj je vse avtomatizirano in tako človeku ni potrebno med delovanjem naprave sodelovati, kar znižuje stroške in pogostost napak.

Težave se pojavljajo pri testiranju naprav, saj mora biti vse zelo natančno odmerjeno in narejeno, sicer ne dobimo pravih rezultatov. Teh težav ni pri veliki laboratorijski opremi, saj je tam manjša možnost napak. Drug problem je grobost površine, saj je lahko površina na mikroravni groba, kar povečuje silo trenja. Trenje zadrži majhne kapljice, ki ostanejo v kanalu, kjer se premikajo snovi. To pripelje do nenadzorovanega mešanja snovi in posledično napačnih rezultatov.

Vse zgoraj našteto pa omogoča hitro analizo in diagnozo snovi na samem mestu. Tako je možna analiza:

- encimov,
- deoksiribonukleinske kisline (DNK),
- virusa človeške imunske pomanjkljivosti (HIV),

[2]Eksotermna reakcija je kemična reakcija, pri kateri se sprošča toplota in ogreva okolico.

- malarije,
- proteinov, povezanih z levkemijo,
- toksičnosti okolja,
- kakovosti vode,
- alkohola in droge v krvi,
- imunosti.

Pri razvoju in uporabi sodelujejo področja, kot so mikroelektronika, elektroavtomatizirano oblikovanje, izdelovalna tehnologija, biokemija, medicina in patologija. Idealen laboratorij na čipu bi bil cenovno ugoden, občutljiv na majhne spremembe, avtomatiziran, integriran in zanesljiv.

Poglavje 2

Mikrofluidni biočipi

V razdelku 2.1 je opisan *zvezno pretočni mikrofluidni biočip*. Sledi rezdelek 2.2, kjer je opisan *digitalni mikrofluidni biočip* (DMFB). Ta razdelek se deli na podrobnejši opis strojnega nivoja v podrazdelku 2.2.1 in opis programskega nivoja digitalnega mikrofluidnega biočipa v podrazdelku 2.2.2.

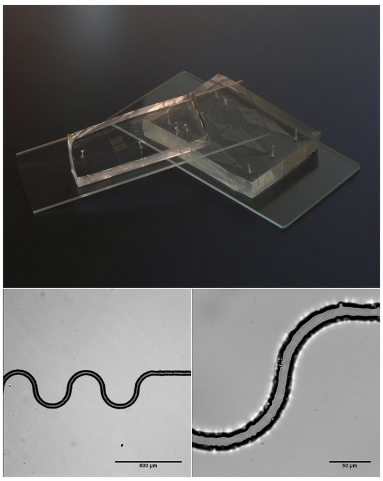
2.1 Zvezno pretočni mikrofluidni biočip

Obstajata dve različni vrsti mikrofluidnega biočipa – *zvezno pretočni* in *digitalni*. Zvezno pretočni je sestavljen iz mikrokanalov, mikročrpalk in mikroventilov. Skozi mikrokanale se manipulira neprekinjen tok različnih snovi. Snov premikamo s pomočjo zunanjih virov, kot so tlačne oziroma mehanske črpalke, mehanske mikročrpalke in kombinacije sile ter elektromehanizmov. Mikroventili skrbijo za usmerjanje tekočine, ki jo premika mikročrpalka.

Za premikanje snovi se uporablja tudi *elektroosmoza*, ki je bila razvita na Univerzi v Michiganu [11]. Elektroosmoza temelji na premikanju ionske raztopine v električnem polju, zato se imenuje tudi elektrokinetska metoda. Kanale, ki temeljijo na premikanju z elektroosmozo, sestavljata dve stekleni plošči, med katerima je tekočina. Na vsaki stekleni plošči se ustvari dvojna plast ionov, ki sestoji iz pozitivno in negativno nabitih elektronov. Ob vzpostavitvi električnega polja se elektroni začnejo premikati in z elektroni se premika tudi tekočina.

Zvezno pretočni mikrofluidni biočip se težko vključuje v različne sisteme, saj ni fleksibilen. Poleg tega trajno jedkane mikrostrukture povzročajo napačno delo-

vanje zvezno pretočnega mikrofluidnega biočipa. Ni primeren za primere, ki zahtevajo visoko stopnjo prilagodljivosti in natančno manipulacijo tekočin. Primeren je za dobro opredeljene in enostavne biokemične preizkuse, kot je kemično ločevanje. Za večjo natančnost pri nadziranju procesa se uporabljajo visoko občutljivi senzorji mikrofluidnih tokov, ki temeljijo na tehnologiji *mikroelektromehanskih sistemov* (MEMS).



Slika 2.1: Zvezno pretočni mikrofluidni biočip Vir: [5].

MEMS [11] je tehnologija majhnih naprav, velikih od 20 mikrometrov do 1 milimetra. Naprave so sestavljene iz dveh komponent – mikroprocesorja in komponent, ki komunicirajo z okolico. Naprave so izdelane iz silikona, keramike, kovin (silicij, zlato, baker, aluminij, titan, volfram, srebro) in polimerov. Slednji se najpogosteje uporabljajo za izdelavo mikrofluidnih naprav. MEMS se uporablja tudi za sistem zračne blazine v avtomobilih, prikazovalne sisteme, senzorje pritiska, mikrofone in merilce pospeška.

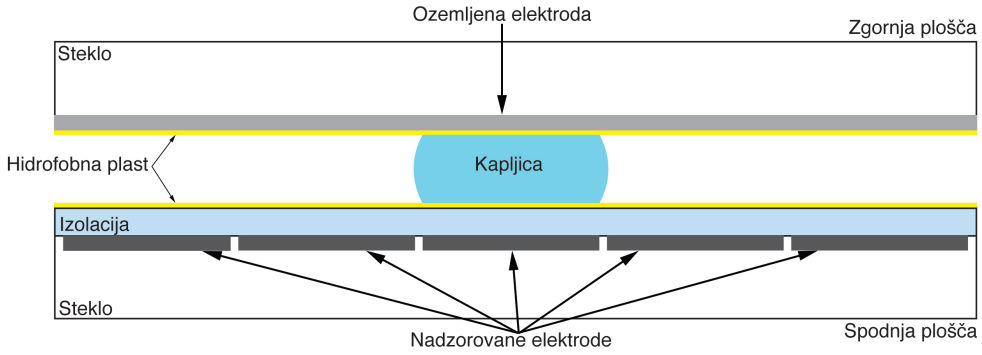
2.2 Digitalni mikrofluidni biočipi

Z razvojem digitalne tehnologije so razvili tudi *digitalni mikrofluidni biočip*. Osnovna ideja je bila zamenjati veliko in drago laboratorijsko opremo z nečim cenejšim, manjšim in z možnostjo prenosljivosti na različna mesta. To pomeni, da je potrebno integrirati različne operacije, ki bi lahko potekale vzporedno na enem čipu.

DMFB je popolnoma digitaliziran, kar omogoča fleksibilne mehanizme za nadzor nad delovanjem in lažjim odkrivanjem ter odpravljanjem napak. Omogoča tudi ponovno postavitev modulov in kapljic med delovanjem, kar pri zvezno pretočnih mikrofluidnih biočipih ni mogoče.

2.2.1 Strojni nivo

DMFB je sestavljen iz dveh glavnih komponent – *rezervoarjev* in *dvodimenzionalnega elektrodnega polja*. Rezervoarji so potrebni za shranjevanje reagentov in vzorcev ter za odpadne snovi, ki nastanejo pri različnih operacijah. Dvodimenzionalno elektrodno polje je sestavljeno iz dveh steklenih plošč, kot je prikazano na Sliki 2.2. Med steklenima ploščama je plast zraka, kjer se gibljejo kapljice snovi. Stekleni plošči sta sestavljeni iz celic, ki vsebujejo pare vzporednih elektrod. Zgornja steklena plošča vsebuje ozemljitveno elektrodo, spodnja steklena plošča pa elektrode, ki so individualno nadzorovane.

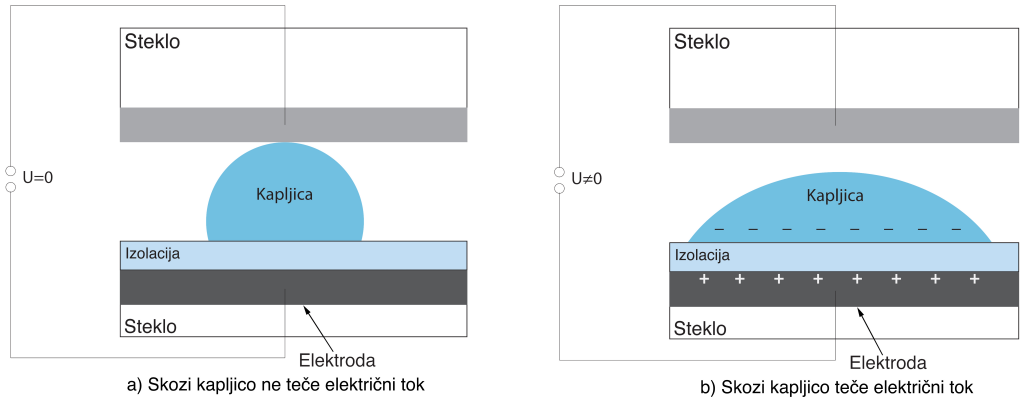


Slika 2.2: Shematski prikaz vertikalnega prereza digitalnega mikrofluidnega biočipa.

Kapljica snovi počiva na hidrofobni[1] površini, ki prekriva notranji del steklene površine. Za premikanje kapljic po elektrodnem polju obstajata dve rešitvi: *dielektroforeza* (DEP) in *elektroomočljivost z dielektričnostjo* (EWOD). DEP za premikanje uporablja visoke frekvence izmenične napetosti, ki dosegajo napetosti med

[1]Hidrofobnost je lastnost nekaterih snovi, ki ne marajo biti v stiku z vodo.

200 in 300 V in frekvenco delovanja med 50 in 300 kHz [11]. Visoke frekvence povzročajo težave pri različnih operacijah, saj se kapljice segrejejo in preizkus se ne izvede v predpisanih pogojih. Zaradi težav z DEP se danes uporablja elektroomočljivost, ki temelji na enosmerni napetosti in se ne segreva tako kot DEP.



Slika 2.3: Prikaz učinka elektroomočljivosti v električnem polju.

Omočljivost je osnova za elektroomočljivost in je značilnost tekočin, da ohranijo stik s podlago. Stopnja omočljivosti je ravnovesje *lepilne sile*, ki omogoča tekočini, da se razširi po podlagi, in *kohezijske sile*, ki odbija kapljico in s tem zmanjša kontaktno površino med kapljico in trdo podlago. Pri veliki lepilni sili je stopnja omočljivosti visoka. Kontaktni kot med kapljico in podlago je manjši od 90^0. Pri kohezijski sili je ravno obratno, saj se kontaktni kot poveča in je večji od 90^0, stopnja omočljivosti je posledično nizka.

Elektroomočljivost je sprememba stopnje omočljivosti z električnim poljem. Pri napetosti 0 V na kapljico deluje kohezijska sila, zato ima kapljica takrat majhno površino, ki se dotika podlage, kot je prikazano na Sliki 2.3 a). Kadar je napetost različna od 0 V, lepilna sila vleče kapljico na elektrodo, zmanjša se kontaktni kot in poveča kontaktna površina med kapljico in elektrodo, Slika 2.3 b).

Pri elektroomočljivosti [15] kapljico premikamo z nadziranjem omočljivosti. To pomeni, če želimo kapljico premakniti na sosednje polje, moramo aktivirati sosednjo elektrodo z uporabo krmilne napetosti in deaktivirati elektrodo pod kapljico, ki jo želimo premakniti. S prilagoditvijo krmilne napetosti, v območju med 0 V in 90 V, nadzorujemo tudi hitrost premikanja kapljic, ki je lahko do 20 cm/s.

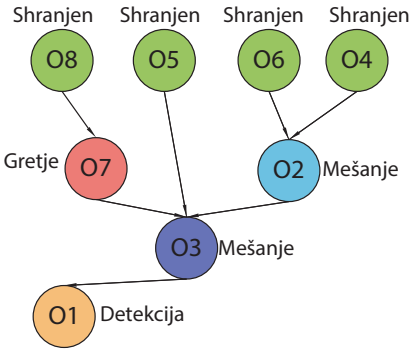
Cilj usmerjanja kapljic je predvsem povezovanje med različnimi moduli in re-

zervoarji v različnih časovnih intervalih.

2.2.2 Programski nivo

Prehod z zvezno pretočnih mikrofluidnih biočipov na DMFB je odprlo možnost programerjem za pisanje svojih algoritmov in operacij, ki jih lahko uporabljajo na DMFB-ju. S tem se je povečala potreba po proizvodnji večjih količin biočipov. Programerji imajo vedno večje zahteve po novih nadgradnjah in manjši velikosti ter porabi.

Za delovanje DMFB-ja je potrebno pred vsakim izvajanjem preizkusa izvesti sintezo. Sinteza je postopek sestavljen iz treh stopenj, ki potekajo na programskem nivoju. Tri stopnje so razvrščanje operacij, postavljanje modulov in usmerjanje kapljic. Vsako stopnjo lahko rešimo z različnimi algoritmi, ki jih napišejo programerji. Po en algoritem za vsako stopnjo je opisan v razdelkih poglavja 3.



Slika 2.4: Sekvenčni graf.

Vsako analizo oziroma preizkus prikazuje usmerjen graf z vozlišči, ki predstavljajo operacije (opisane v nadaljevanju), in povezavami, ki predstavljajo odvisnosti med operacijami. Za vsak preizkus je podana tudi velikost samega elektrodnega polja, ki ga bomo uporabili za izvajanje preizkusa, in maksimalen dovoljen čas trajanja preizkusa.

Moduli

Za vsak modul je določena dolžina in širina, ki jo potrebuje, da se izvede, ter potreben čas izvajanja. Moduli, kot so mešalnik, razdruževalnik in shranjevalnik,

niso fiksne naprave kot pri zvezno pretočnih mikrofluidnih biočipih, kjer so posebni čipi (mikromešalnik, mikrokomora) pritrjeni na podlago. Pri DMFB-ju so moduli kot navidezne naprave, ki se tvorijo kar na samem elektrodnem polju. Za določen modul se izbere potrebno število elektrod na elektrodnem polju, ki jih v določenem času lahko uporablja samo ena operacija. Vsak preizkus lahko vsebuje različne module, kot so:

Mešalnik ima nalogo zmešati dve kapljici v eno. Vsebuje dvoje vhodnih vrat za dve kapljici in ena izhodna vrata. Po vstopu kapljic v modul se kapljici združita v eno kapljico. Kapljica se nekaj časa premika znotraj modula, da se snov popolnoma zmeša. Nato se kapljica postavi na mesto za izhod in izstopi, ko poteče predvideni čas.

Razdruževalnik ima nalogo razdružiti eno kapljico v dve. Vsebuje ena vhodna vrata in dvoje izhodnih. Kapljica ob predvidenem času vstopi v modul in se razdeli na dve kapljici. Vsaka izmed kapljic se postavi na izhod in ob končnem času izstopi iz modula.

Shranjevalnik ima nalogo shranjevanja kapljic, ki jih trenutno ne potrebujemo. Shranjevalnik ima lahko več vhodnih in izhodnih vrat, ki so odvisna predvsem od velikosti modula. Kapljice ob začetku vstopijo in nato mirujejo znotraj modula toliko časa, kot je bilo predvidenega za shranjevanje. Ob poteku časa kapljice izstopijo iz modula.

Rezervoar hrani snovi pred vstopom na elektrodno polje (vhodni rezervoar) in po izstopu z elektrodnega polja (izhodni rezervoar). Vsebuje ena vhodno-izhodna vrata, kjer kapljica vstopi in izstopi.

Optični detektor ima nalogo pridobivati podatke o kapljicah. Vsebuje ena vhodna in izhodna vrata. Kapljica vstopi v modul in se postavi pod detektor kjer miruje nekaj časa, da lahko optični detektor pridobi potrebne podatke. Pred koncem se postavi na izhod in ob koncu izstopi iz modula.

Grelnik ima nalogo segrevanja kapljice do določene temperature. Grelnik ima tako kot detektor ena vhodna in ena izhodna vrata. Ob začetku kapljica vstopi in se postavi na grelnik, kjer stoji toliko časa, kot ga potrebuje da

kapljico segreje na predvideno temperaturo. Ob koncu zapusti modul skozi izhodna vrata.

Fotodioda s svetlečo diodo (LED) ima nalogo merjenja koncentracije snovi v kapljici s pomočjo pretvarjanja svetlobe v električni tok.

Moduli se dinamično prilagajajo, saj se lahko zgodi, da posamezna elektroda ne deluje pravilno in modul se mora prestaviti na drugo mesto. Vsak modul ima natančno definirane vhode in izhode, kar preprečuje, da bi se kapljice med seboj zmešale, ko izstopijo iz modula.

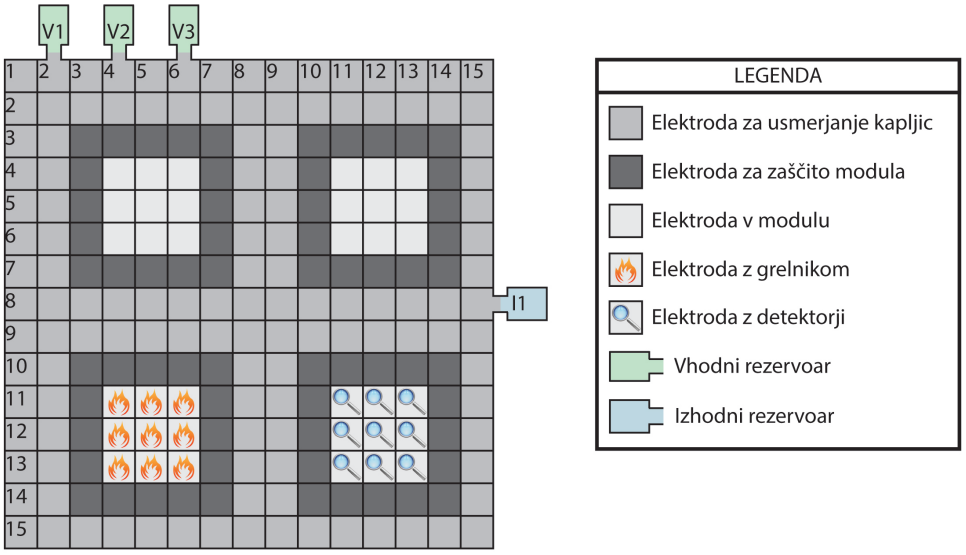
V primeru, da modula in kapljice v naslednjem koraku ne potrebujemo, se lahko modul spremeni v shranjevalnik in kapljica počaka znotraj njega. Ena od možnosti je tudi, da kapljico potrebujemo za naslednjo operacijo v istem modulu in zato se mora le premakniti znotraj modula na mesto, kjer kapljice vstopijo.

Moduli, kot so rezervoar, optični detektor, grelnik in fotodioda s svetlečo diodo, so izdelani in vgrajeni že v tovarni in jih ni mogoče prestavljati. Po potrebi jih lahko vključimo ali izključimo in površino uporabimo za druge module.

Izhodni podatki algoritmov za postavitev modulov, omogočajo grafični prikaz modulov na elektrodnem polju. Ta prikaz je lahko v 2D ravnini ali 3D prostoru, kjer vodoravni osi predstavljata elektrodno polje in tretja navpična os predstavlja čas. Tako za vsak modul nastane en kvader, kjer spodnja ploskev kvadra predstavlja velikost modula na elektrodnem polju in višina kvadra, ki predstavlja čas delovanja modula.

Problem postavljanja modulov nastane zaradi časa delovanja določenega modula. Na primer, mešanje lahko traja med 5 in 10 s, medtem ko premik kapljice med moduli ali do rezervoarja traja samo 5 do 10 ms. Razlika v trajanju pomeni, da moramo kapljice med operacijami shraniti, kar zavzame prostor na elektrodnem polju.

Shematski prikaz DMFB-ja s postavljenimi moduli in rezervoarji je predstavljen na Sliki 2.5. Z zeleno barvo so levo zgoraj predstavljeni vhodni rezervoarji, v katerih se hranijo reagenti in vzorci, preden jih premaknemo na elektrodno polje. Izhodni rezervoar je na desni strani elektrodnega polja predstavljen z modro barvo. Elektrodno polje je sestavljeno iz petih različnih elektrod. Elektrode za usmerjanje kapljic so namenjene usmerjanju kapljic po elektrodnem polju. Elektrode za zaščito modula ščitijo modul pred vdorom zunanjih kapljic oziroma nenamernemu



Slika 2.5: Shematski prikaz DMFB.

izstopu kapljic iz modula. Te elektrode ne smejo biti aktivirane med delovanjem modula. Elektrode v modulu so namenjene izvajanju različnih operacij, ki so opisane v nadaljevanju. Elektroda z grelnikom ima vgrajen grelnik, ki segreje kapljico do določene temperature. Elektroda z detektorjem ima vgrajen detektor, ki pridobi podatke o kapljici.

Operacije

Operacija je proces, ki za svoje delovanje uporablja modul. Rezultat operacije je odvisen od izbire in delovanja modula. Po končanju nekaterih operacij je potrebno površino elektrod tudi sčistiti in odstraniti ostanke. Te odpadke shranimo v rezervoar za odpadke, kar vzame nekaj časa in med tem ne moremo nadaljevati z operacijami. Temu se lahko delno izognemo s plastjo silikonskega olja, ki preprečuje, da bi kapljice puščale sled, a še zmeraj ne moremo preprečiti kontaminacije[2] vseh snovi. Operacije lahko razdelimo tudi na posamezne dejavnosti, kot so:

- točenje – kapljica se prelije iz rezervoarja na čip,

[2]Kontaminacija je prisotnost nezaželjene snovi ali delcev v drugi snovi.

- usmerjanje – premikanje kapljic znotraj elektrodnega polja,
- mešanje – dve kapljici se združita v eno kapljico,
- razdruževanje – kapljica se razdeli na dve manjši kapljici,
- redčenje – kapljica se razredči z kapljico topila, običajno v mešalniku,
- zaznavanje – kapljica je na mestu z različnimi detektorji, ki detektirajo različne lastnosti kapljice,
- gretje – kapljico se segreje na predvideno temperaturo,
- čiščenje – po prenosu kapljice na elektrodnem polju ostanejo sledi, ki se odstranijo z čiščenjem.

Usmerjanje kapljic na elektrodnem polju zahteva preračunavanje poti vsake kapljice posebej in to vedno, ko pride do spremembe v delovanju. Problemi lahko nastanejo z nenamernim mešanjem različnih kapljic, ki pridejo na isto elektrodo. Vsaka kapljica mora imeti tudi možnost, da pride do vsakega modula oziroma rezervoarja ne glede na njen položaj in čas. Za reševanje problemov z nenamernim mešanjem kapljic in možnostjo dostopa do cilja obstajajo različni algoritmi.

Naslavljanje elektrod

Vsaka elektroda je v osnovi individualno nadzorovana, zato je treba pošiljati signale za vsako elektrodo posebej. To pomeni daljši čas premikanja in večjo porabo energije. Zato so razvili še dve metodi naslavljanja elektrod – *navzkrižno naslavljanje* in *naslavljanje pinov* [9].

Prva metoda, navzkrižno naslavljanje (angl. cross-reference), deluje na način, da se izbereta vrstica in stolpec, kjer je potrebno aktivirati elektrodo. Za $M \times N$ veliko polje, kjer je M število stolpcev in N število vrstic, potrebujemo toliko pinov, kolikor je vrstic in stolpcev skupaj, torej $M + N$. Elektrodo na mestu (m, n) aktiviramo tako, da pošljemo signal stolpcu m in vrstici n. Ta metoda omogoča reprogramiranje elektrod, ki je tudi kompleksna in zahteva nekaj časa.

Druga metoda, naslavljanje pinov (angl. pin-constrain), naslavlja več elektrod hkrati. Predstavlja velik napredek, saj se zmanjša število nadzornih pinov tako, da je več elektrod hkrati nadzorovanih z enim kontrolnim pinom. Težave se pojavijo,

če želimo spremeniti, katere elektrode naj se naslavljajo skupaj, saj skupin elektrod ne moremo spremeniti. Pomembno je, da združimo pravilne elektrode skupaj, da ne pride do konfliktov med signali ali mešanja kapljic.

Poglavje 3

Sinteza

Za delovanje DMFB-ja so potrebni različni algoritmi. Seveda se algoritmi med seboj razlikujejo in razvijajo se novi, boljši, ki izvedejo preizkus hitreje. Številni DMFB-ji se uporabljajo za varnostno pomembne aplikacije, kot je zdravje pacientov, oskrba novorojenčkov in spremljanje toksinov v okolju.

V nadaljevanju sledi splošen pogled na celoten postopek sinteze v razdelku 3.1. Sledijo podrobnejše predstavitve problemov, ki smo jih srečali pri sintezi. Pri vsakem problemu je opisan še en algoritem. Prvi je problem razvrščanja operacij v razdelku 3.2 z algoritmom razvrščanje poti. Sledi predstavitev postavljanja modulov na elektrodno polje v razdelku 3.3 in algoritma na osnovi simuliranega ohlajanja. Problem usmerjanja kapljic na elektrodnem polju je skupaj z algoritmom predstavljen v razdelku 3.4. Na koncu je v razdelku 3.5 predstavljen še simulator za simuliranje DMFB-ja.

3.1 Celostni pogled na sintezo

Sinteza je programska priprava preizkusa, da se bo preizkus lahko izvedel na DMFB-ju. Sestavljena je iz več stopenj, ki so potrebne, da pridemo do končnega rezultata. Po končanju izvajanja vsake stopnje, algoritem izdela vmesno rešitev, ki jo naslednja stopnja uporabi za svoje delovanje.

Vhodni podatki za sintezo morajo biti podani za vsak preizkus. Prvi podatek je *sekvenčni graf* (angl. sequencing graph) $G = (V, E)$, kjer so V vozlišča, ki predstavljajo operacije, in E usmerjene povezave oziroma odvisnosti med opera-

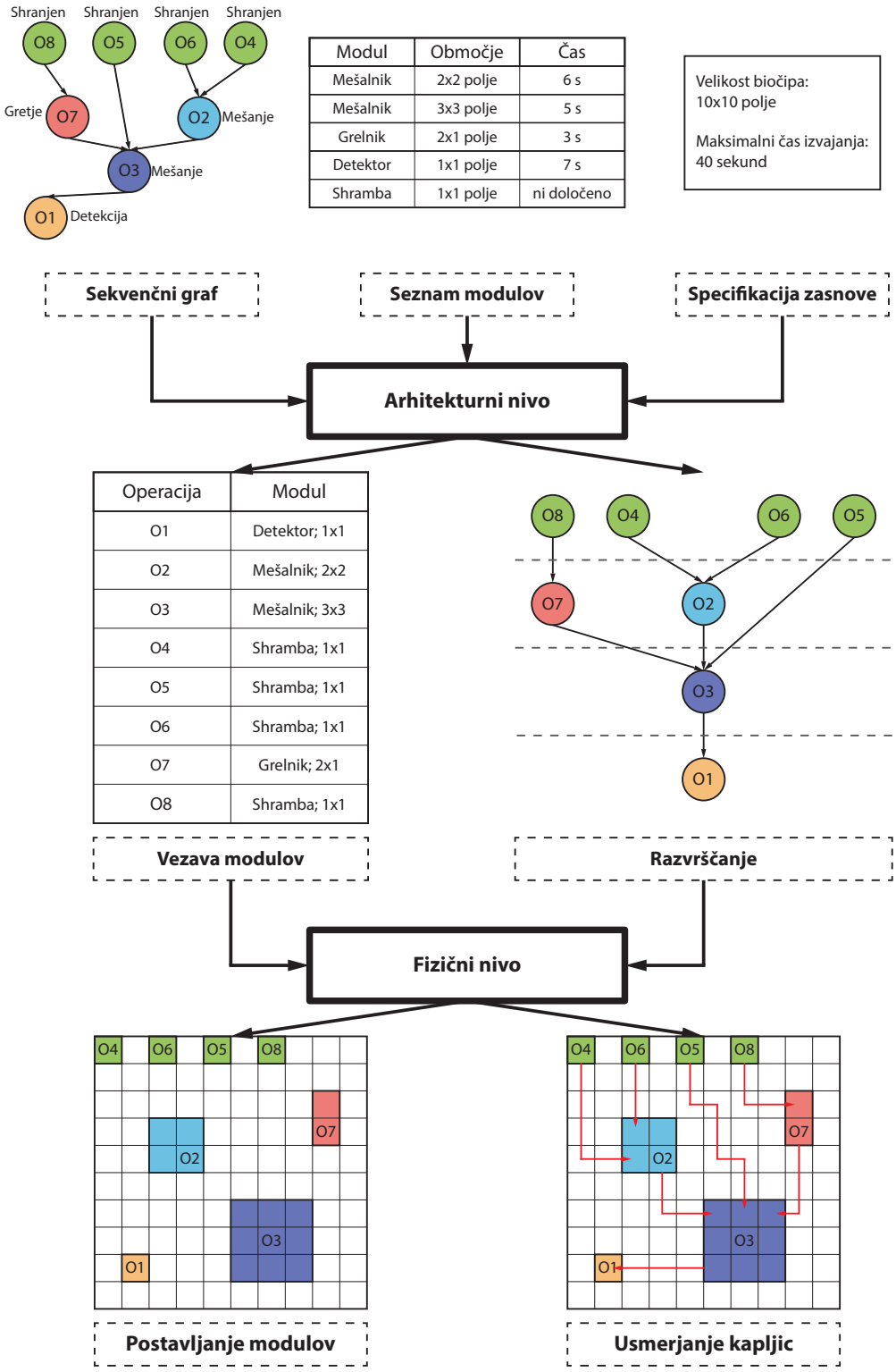
cijami. Primer sekvenčnega grafa je na Sliki 3.1 levo zgoraj, kjer vozlišča od $O1$ do $O8$ predstavljajo operacije, povezave predstavljajo odvisnosti med operacijami. Drugi podatek je seznam modulov, ki jih imamo možnost izbrati za operacije. Za vsak modul je podano, koliko prostora zavzame med delovanjem in potreben čas za izvedbo operacije. Na Sliki 3.1 je prikazan primer seznama modulov na sredini zgoraj. Zadnji vhodni podatek je specifikacija zasnove elektrodnega polja in zahteve za izvedbo. Podana je velikost samega elektrodnega polja in največji dovoljen čas za izvedbo preizkusa. V nekaterih preizkusih je definirano tudi, ali DMFB omogoča čiščenje elektrod. Sintezo delimo na dva nivoja: *arhitekturni* in *fizični*.

3.1.1 Arhitekturni nivo

Na arhitekturnem nivoju, se vsem operacijam določi module, ki jih potrebujejo operacije za delovanje. Na primer, na Sliki 3.1 je prikazan primer vezave modula pod arhitekturnim nivojem levo, kjer se operaciji $O3$ dodeli mešalnik z velikostjo 3×3 in časom delovanja 5 s. Sledi razvrščanje operacij na osnovi sekvenčnega grafa in časa, dodeljenega operaciji z vezavo modula. Kot rezultat dobimo nov sekvenčni graf z razvrščenimi operacijami. Primer razvrščanja je prikazan na Sliki 3.1 pod arhitekturnim nivojem desno. Za arhitekturnim nivojem sledi fizični nivo, v katerem sekvenčni graf, pridobljen z razvrščanjem, uporabimo za postavljanje modulov na elektrodno polje.

3.1.2 Fizični nivo

Fizični nivo je zelo pomemben del, saj moramo poskrbeti, da se moduli med seboj ne prekrivajo in se operacije izvajajo vzporedno. Vedeti moramo, da imamo še tretjo dimenzijo, in sicer čas, skozi katerega se moduli lahko odstranijo ali dodajo novi. Pomembno je, da zagotovimo dovolj prostora med moduli, kjer se bodo kapljice lahko premikale. Po drugi strani moramo zmanjšati uporabljeno površino na najmanjšo možno. Primer postavitve modulov na elektrodno polje je levo spodaj na Sliki 3.1. Po končanem postavljanju modulov sledi usmerjanje kapljic na elektrodnem polju. Poskrbeti moramo, da vsako kapljico pripeljemo od njenega izvora do cilja, ne da bi se zmešala s kapljico za katero mešanje v preizkusu ni predvideno. Na Sliki 3.1 je desno spodaj prikazan primer usmerjenih kapljic in



Slika 3.1: Potek sinteze na DMFB.

njihovih poti. Na DMFB-ju, ki omogoča čiščenje elektrod, moramo usmeriti tudi čistilne kapljice, ki sčistijo sledi kapljic.

Izhod sinteze so podatki za postavitev modulov in usmerjanje kapljic na elektrodnem polju. Izhodne podatke lahko uporabimo na simulatorju ali pravem DMFB-ju, kjer se izvedejo vse operacije v predvidenem zaporedju in dajo končni rezultat preizkusa.

3.2 Razvrščanje operacij

3.2.1 Opis problema

Pri vsakem preizkusu se najprej izvede *vezava modulov* in *razvrščanje operacij*. Problem razvrščanja operacij že vsebuje problem vezave modulov. Učinkovito razvrščanje ima izmed vseh stopenj sinteze največji vpliv na čas izvajanja preizkusa. Vpliva tudi na to, ali se bodo podatki, ki jih dobimo po končanih algoritmih za postavljanje modulov in usmerjanje kapljic, lahko uporabili za izvedbo preizkusa. Cilj razvrščanja operacij je predvsem razvrstiti operacije tako, da se zmanjša čas izvajanja in poveča število vzporedno delujočih modulov.

Za razvrščanje operacij potrebujemo sekvenčni graf z operacijami in odvisnostmi med njimi. Zagotoviti in upoštevati moramo omejitve, ki so podane v sekvenčnem grafu. Za vsako operacijo, ki jo razvrstimo, moramo zagotoviti, da obstaja prost modul za izvedbo operacije. Modul lahko ob določenem času uporablja samo ena operacija. Pri nekaterih preizkusih lahko obstajajo različni moduli, ki jih določena operacija lahko uporabi.

Čas izvajanja določene operacije, kot sta mešanje in razdruževanje, je odvisen od velikosti modula, ki ji je dodeljen. Večji kot je modul, manj časa bo potrebnega za izvedbo operacije, manjši je modul, več časa bo potrebnega za izvedbo. Vendar je problem ravno v velikosti, saj ne moremo na elektrodno polje postaviti en velik modul, kar bi pomenilo slabo izkoriščenost vzporednega delovanja, več majhnih pa bi povečalo čas izvajanja preizkusa.

Naloga razvrščevalnika je tudi določevanje shranjevalnih modulov, ki hranijo kapljice, medtem ko jih ne potrebujemo. Rešitev razvrščanja vsebuje razvrščene operacije z moduli in začetne ter končne čase izvajanja.

3.2.2 Delovni moduli

Algoritem razdeli elektrodno polje na območja, ki se imenujejo delovni moduli. Delovne module delimo na splošne module in module za posebne namene. V splošnih modulih se izvajajo operacije mešanja, razdruževanja in shranjevanja. Moduli za posebne namene so splošen modul opremljen s senzorjem, grelcem ali zunanjo napravo. Delovni moduli so postavljeni tesno skupaj, vendar je med njimi dovolj prostora za transport kapljic. Število vseh delovnih modulov je N_m, vseh splošnih N_g in vseh posebnih modulov tipa i je N_i. Iz tega sledi, da je število vseh delovnih modulov enako:

$$\sum_{\forall i} N_i + N_g \quad = \quad N_m.$$

V času, ko katerikoli delovni modul ne izvaja operacije, lahko postane modul za shranjevanje največ s_c kapljic. Naj bo sp_{it} število posebnih operacij tipa i, ki se izvajajo v času t, g_t število splošnih operacij, ki se izvajajo v času t, in s_t število shranjenih kapljic v času t. Iz tega sledita neenakosti, ki morata veljati:

$$\sum_{\forall i} sp_{it} + g_t + \lceil s_t/s_c \rceil \quad \leq \quad N_m, \forall t, \tag{3.1}$$

$$sp_{it} \quad \leq \quad N_i, \forall i, \forall t. \tag{3.2}$$

Neenakost 3.1 mora veljati ob vsakem času izvajanja preizkusa, saj vsota vseh posebnih operacij za vse tipe i, vseh splošnih modulov in razmerja med številom shranjenih kapljic ter številom vseh kapljic, ki jih lahko shranimo ne sme presegati števila vseh delovnih modulov. Neenakost 3.2 pa mora, enako kot prejšnja, veljati za vsak čas izvajanja preizkusa, kjer število posebnih operacij tipa i ne sme biti večje od števila razpoložljivih posebnih modulov tipa i.

3.2.3 Prioriteta

Funkcija prioritete se uporablja za določevanje prioritet operacijam za lažje razvrščanje in odločanje o prednosti operacij. Algoritem uporablja dve prioriteti, da zmanjša čas prisotnosti kapljice na elektrodnem polju. Zgoden vstop kapljice na elektrodno polje algoritem prepreči s *prioriteto neodvisnih poti* (IPP), ki jo izračunamo vsakemu vozlišču na osnovi števila neodvisnih poti. Število neodvisnih poti predstavlja število vseh kapljic, ki po koncu operacije izstopijo iz modula.

Začetno vozlišče v preizkusu je znano kot *vodja poti*, saj iz njene operacije ponavadi izstopi le ena kapljica. Pot vodje poti na začetku predstavlja samo eno vozlišče, nato mu dodajamo z njim povezana vozlišča z najvišjo vrednostjo IPP. Ta postopek ponavljamo, dokler ne pridemo do operacije mešanja ali izhodnega rezervoarja.

Za vsako vozlišče izračunamo tudi *prioriteto kritične poti* (CPP), kot dolžino najdaljše časovne poti med trenutnim vozliščem in izhodnim rezervoarjem. Ta prioriteta nam koristi, ko imata dve vozlišči enak IPP in izberemo tistega z nižjim CPP. Tako dobimo rešitev s krajšim časom delovanja.

Na primer, da imamo dve kapljici, ki želita ob enakem času uporabiti modul (z grelnikom ali detektorjem), mi pa imamo na elektrodnem polju samo enega, moramo eno kapljico shraniti v splošen modul, dokler druga ne konča z operacijo. S tem, ko izberemo kapljico, ki ima krajšo pot do modula, zmanjšamo čakalni čas kapljice v shrambi in povečamo izkoriščenost sistema.

3.2.4 Razvrščanje poti

Razvrščanje poti [8] je hevristični razvrščevalni algoritem, ki razvršča zbirko povezanih odvisnih operacij in ne vsakega vozlišča sekvenčnega grafa posebej. Odvisne operacije so vozlišča sekvenčnega grafa povezana s povezavami, ki imajo skupno vozlišče, kjer se izvede operacija mešanja. Razvrščanje poti poveča časovno in prostorsko izkoriščenost elektrodnega polja in izdelavo boljših načrtov za dane preizkuse.

Primeren je tudi za sprotno sintezo (angl. online), vendar je treba natančno definirati preizkus, da deluje pravilno. Sprotna sinteza pomeni, da algoritem dobiva podatke sproti med delovanjem in ne na začetku vseh podatkov naenkrat. To omogoča hitrejše delovanje, saj ni treba čakati na podatke.

V Algoritmu 1 moramo najprej vsem vozliščem v sekvenčnem grafu določiti obe prioriteti (IPP in CPP). Določiti moramo kandidate operacij, katerih starši so vhodni rezervoarji. Vrstice od 5 do 29 ponavljamo dokler ne razvrstimo vseh kandidatov operacij. V vrsticah od 6 do 8 izberemo vodjo poti z najnižjo prioriteto IPP, v primeru enakosti pa najnižji CPP in ponastavimo čas t in pot P. Med delovanjem algoritma poskušamo celotni poti, od vodje poti do operacije izhodnega rezervoarja oziroma mešanja, dodeliti module, ki jih operacije potrebujejo za svoje izvajanje.

Algoritem 1: Razvrščanje poti

```
Sekvenčni graf G = (V, E)
Omejitve modulov N_g, N_i in s_c
Vsem vozliščem v ∈ V se določita prioriteti IPP in CPP
Kandidati operacij R = {v_i ∈ V
  : TipModula(v_j) == vhodni rezervoar, ∀j:(v_j, v_i) ∈ E}
repeat
    Izberemo S ⊆ R: Prioriteta(S) ≤ Prioriteta(r), ∀r : r ∈ R
    Časovni korak t = 1
    Pot P = ∅
    repeat
        Poskušamo poiskati najzgodnejši prost časovni korak t_i in
        splošen ali poseben modul k za S : ProstModul(ts, k) == true,
        ∀ts : t_i ≤ ts ≤ t_i + S.trajanjeOperacije medtem, ko veljata
        enačbi 3.1 in 3.2
        if Najdemo prost čas za S then
            Nastavi S.začetek = t_i
            Nastavi S.tipModula = k
            Dodaj S v P
        else
            Nastavi P.razvrščujoč = false
        end if
        Izberi nov S ⊆ S.otrok : Prioriteta(S) ≤ Prioriteta(S_ch), ∀S_ch :
        S_ch ∈ S.otrok
    until ((TipModula(S) == mešalnik) ∧ (S.razvrščen == false)) ∨
    (TipModula(S) == izhodni rezervoar) ∨ (P.razvrščujoč == false)
    if P.razvrščujoč == true then
        for ∀p ∈ P do
            Nastavi p.razvrščen == true
            Shrani module za pot P
            for ∀c : c ∈ p.otrok ∧ c ∉ P do
                Dodaj c med kandidate operacij R
            end for
        end for
    end if
until Vsi kandidati operacij so razvrščeni: R = ∅
```

Algoritem poskuša, v vrstici 10, za trenutno vozlišče S poiskati čas, ki še ni izkoriščen in kjer bi lahko vstavili vozlišče s prostim modulom. Če najde neizkoriščen čas, pomeni, da obstaja splošen ali poseben modul, ki je prost v časovnem intervalu od t_i do $t_i + S.trajanjeOperacije$. Pomembno je tudi, da so omogočeni potrebni vhodni rezervoarji, če so vozlišča začetna, da dobijo potrebne vzorce in reagente. Začeten čas operacije in potreben modul za vozlišče S se začasno shranijo. Vozlišče S v vrstici 14 dodamo trenutni poti P.

Naslednje vozlišče, ki je otrok vozlišča S, izberemo v vrstici 18, na osnovi prioritete IPP. Če imata dva otroka vozlišča S enako vrednost IPP izberemo tistega z nižjo vrednostjo CPP. V primeru, da je novo vozlišče S izhodni rezervoar ali mešalna operacija, ki še ni bila razvrščena ($S.razvrščen$), kar preverimo v vrstici 19, pomeni, da je P zaključena pot in jo lahko označimo za razvrščeno. Če novo vozlišče S ni izhodni rezervoar ali nerazvrščena mešalna operacija, ga poskušamo dodati v pot P. Pot, ki smo razvrstili odstranimo iz seznama poti za razvrščanje, njene module, ki so bili začasno shranjeni, pa označimo kot zasedene. Vsi nerazvrščeni otroci poti P so dodani v seznam kandidatov operacij. Povezavo med potjo in kandidatom operacije predstavlja kapljica, ki jo je potrebno shraniti, dokler jo ne bomo razvrstili. Pomembno je, da jo shranimo in se ne meša z drugimi, saj je nekaj časa ne bomo potrebovali. Kapljico dodamo na seznam kapljic, ki so izstopile iz že usmerjenih poti staršev. Te kapljice odstranimo iz seznama, ko jih uporabimo pri razvrščanju in je njihova pot uspešno označena kot razvrščena.

Poti P ne moremo usmeriti, če algoritem ne najde neizkoriščenega časa, kjer bi lahko vozlišče S dodali poti P, ali zaradi zasedenih potrebnih modulov. Pot, ki jo trenutno ne moremo razvrstiti ($P.razvrščujoč$), označimo v vrstici 16 . Prav tako sprostimo vse module, ki smo jih označili kot uporabljene. Takšno pot bo algoritem poskušal razvrstiti pozneje. Zdaj bo poskusil razvrstiti eno izmed preostalih še nerazvrščenih poti.

Algoritem se konča, ko razvrstimo vse poti. Rezultat algoritma je urejen graf operacij s potrebnimi moduli in začetnim ter končnim časom vsake operacije posebej. Dodani so tudi shranjevalni moduli, kjer je potrebno shraniti kapljico za določen čas.

3.3 Postavitev modulov

3.3.1 Opis problema

Problem *postavljanja modulov* na elektrodno polje je NP-težek [10]. Reševanje problema je pomembno za hitrejšo analizo preizkusov, saj omogoča izvajanje več operacij vzporedno. Je tudi eden glavnih problemov fizične zasnove DMFB-ja. Glavni cilj je poiskati lokacijo na elektrodnem polju za različne module ob različnih časih. Glede na čas se lahko tudi menjajo moduli, tako da je najprej na nekem mestu mešalnik, po končanju operacije lahko na isto mesto postavimo razdruževalnik. Potrebno je upoštevati tudi začetni in končni čas za vsako operacijo ter poskrbeti za pravočasno shranjevanje neuporabljenih kapljic. Rezultat postavljanja modulov je položaj vsakega modula na elektrodnem polju za vsako operacijo ob času določenem v sekvenčnem grafu. Med moduli moramo zagotoviti dovolj prostora za usmerjanje kapljic.

Postavljanje modulov na elektrodno polje si lahko predstavljamo kot postavljanje zabojev v 3D prosotru. 3D prostor sestavlja 2D elektrodno polje in čas. *Problem 3D pakiranja* (angl. 3D packing problem) je razmestiti dane zaboje različnih velikosti v prostor tako, da pokrijemo majhno površino elektrodnega polja, a dopuščamo vzporedno postavljanje. Cilj je optimizacija porabljenega prostora na elektrodnem polju in časa za izvajnaje preizkusa. Posledica majhnega števila aktivnih elektrod je manjša poraba energije, kar zmanjša stroške izvajanja preizkusa.

Problem nastane tudi ob nedelovanju oziroma napačnem delovanju določene elektrode, kar je treba upoštevati, da ne pride do napačnih rezultatov. Pri rekonfiguracijskih[1] modulih se ta problem lahko reši s prestavitvijo modula na drugo mesto. Pri nerekonfiguracijskih[2] modulih problema ne moremo odpraviti, kar lahko pripelje do nedelujočega DMFB-ja.

[1]Rekonfiguracijaki so moduli, ki jih lahko premikamo po elektrodnem polju (mešalnik, shranjevalnik, razdruževalnik).

[2]Nerekonfiguracijski so moduli, ki jih ne moremo premikati po elektrodnem polju, saj so bili vgrajeni že v tovarni (grelnik, optični detektor, rezervoar, fotodioda).

3.3.2 Pristopi reševanja

Problem postavitve modulov na DMFB-ju lahko rešimo z dvema različnima pristopoma [10] – *posredni* in *neposredni pristop*.

Posredni pristop Posreden pristop (angl. indirect approach) temelji na abstraktni predstavitvi postavitve modulov, kar pomeni, da se moduli samo navidezno postavijo na elektrodno polje. Proces kodiranja pridobi sekvenčni graf, s katerim opiše topološka razmerja med moduli. Nato sledi proces postavljanja modulov s prej opisanimi topološkimi razmerji med moduli.

Obstajajo različne metode, kot so zaporedje trojčkov (angl. sequence triplet), 3-D-sub TCG in T-tree, ki za reševanje problemov uporabljajo posredni pristop. Te metode imajo to prednost, da so njihovi vmesni rezultati izvedljivi in s tem optimizirajo iskanje prostora za postavitev modula.

Neposredni pristop Pri neposrednem pristopu (angl. direct approach) se moduli postavljajo na fizične koordinate elektrodnega polja s svojo velikostjo in orientacijo. Zaradi neposrednega postavljanja modulov na elektrodno polje prihaja do prekrivanja različnih modulov med seboj, kar pripelje do kaznovanja rešitve, ki se odraža v kriterijski funkciji. Kriterijska funkcija je pomembna za ocenjevanje vrednosti rešitve danega problema. Prednost neposrednega pristopa je podrobna geometrična informacija o postavitvi modulov na elektrodno polje. Dobimo jo s postavljanjem modulov na elektrodno polje.

3.3.3 Simulirano ohlajanje

Simulirano ohlajanje (SA) je metahevristična metoda za iskanje globalnega optimuma. Metoda zmeraj najde približek globalnega optimuma, lahko najde tudi optimalno rešitev, na velikem iskalnem prostoru. Metodo so Scott Kirkpatrick, C. Daniel Gelatt in Mario P. Vecchi prvi opisali leta 1983 [14], nato še neodvisno Vlado Černý leta 1985 [7].

SA izhaja iz žarjenja in ohlajanja različnih kovin pri njihovi predelavi. V metodi SA se uporablja princip počasnega ohlajevanja, kot počasno zmanjševanje verjetnosti sprejema slabe rešitve. SA je dobro raziskana optimizacijska metoda, široko uporabljena za probleme postavljanja modulov. Uporablja se tudi za druge

kombinatorične probleme, kot je problem trgovskega potnika. Metoda preiskuje prostor rešitev optimizacijskega problema in omogoča iskanje lokalnega optimuma s sprotnim izboljševanjem trenutne rešitve.

Vsi parametri v SA so določeni eksperimentalno med ohlajanjem. Temperaturo T v vsakem koraku zmanjšujemo po formuli

$$T_{nov} = \alpha \times T_{star},$$

kjer je α faktor ohlajanja ($\alpha = 0.8$). Algoritem se konča, ko se T zmanjša na vrednost $T = 0$. Število iteracij izboljšav N za dano T določimo po formuli

$$N = N_a \times N_m,$$

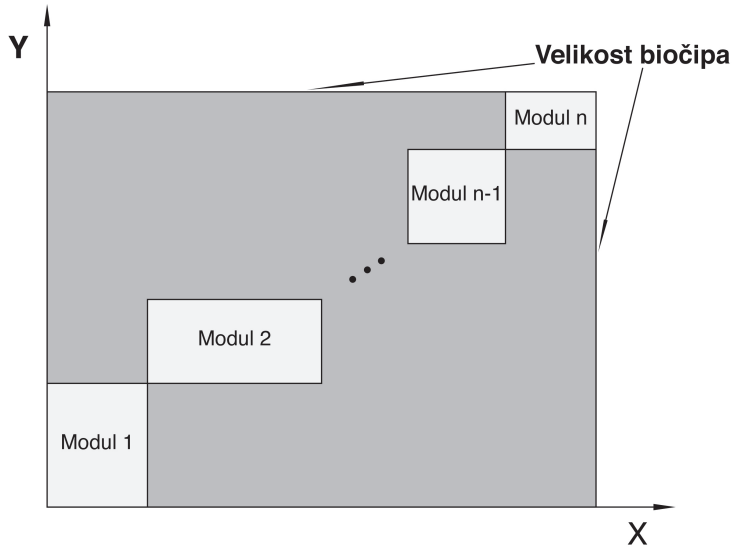
kjer je N_a konstanta, ki jo določimo eksperimentalno, N_m je število modulov, ki se uporabljajo v preizkusu.

3.3.4 Postavljanje modulov

Algoritem temelji na neposrednem pristopu, opisanem v podrazdelku 3.3.2. Začetna postavitev modulov za SA ni tako pomembna oziroma ima majhen vpliv na končen rezultat. Module na začetku postavimo na elektrodno polje po diagonali, tako da se ne prekrivajo in so znotraj elektrodnega polja. Začetno postavitev modulov prikazuje Slika 3.2. Med samim izvajanjem algoritma je treba preprečevati, da bi bili moduli postavljeni izven velikosti elektrodnega polja. Na začetku se nastavi temperatura T na veliko vrednost, kar omogoča, da je sprejeta vsaka nova prestavitev modula na elektrodnem polju.

SA ima različne funkcije za prestavljanje modulov na elektrodnem polju. Vsakemu modulu je ob vsakem nižanju temperature T, skozi proces ohlajanja, dodeljena verjetnost p, da bo zamenjal lokacijo, ali verjetnost $1 - p$, da se bosta zamenjala naključna dva modula med seboj. Za vsako verjetnost se lahko izvede ena izmed sledečih funkcij. Izbran je lahko naključen modul, ki se prestavi na naključno izbrano mesto. Modulu lahko med prestavljanjem tudi spremenimo orientacijo in ga nato postavimo na novo lokacijo.

Položaje lahko z verjetnstjo $1 - p$ izmenjata dva naključna modula med seboj, nobenemu, enemu ali obema lahko naključno spremenimo tudi orientacijo. Učinkovitost razmerja $p/(1 - p)$ je določena eksperimentalno.



Slika 3.2: Začetna postavitev modulov na DMFB-ju.

Vsak modul je nadzorovan z nadzornim oknom, saj sprememba lokacije, še posebej, če je razdalja velika, poveča čas delovanja. Pri nizkih temperaturah med procesom ohlajanja imajo samo moduli, ki še niso bili prestavljeni, možnost, da bodo premaknjeni. Ostali moduli pri nizkih temperaturah ne bodo prestavljeni na veliko razdaljo, saj se dovoljena razdalja premika z nižanjem temperature manjša.

3.4 Usmerjanje kapljic

3.4.1 Opis problema

Glavni cilj *usmerjanja kapljic* na elektrodnem polju je pripeljati kapljico od njenega izvora do cilja. Ta problem lahko predstavimo z iskanjem poti v acikličnem usmerjenem grafu. Druga možnost je, da problem predstavimo kot iskanje poti v 3D prostoru, tako kot pri postavljanju modulov na elektrodno polje. Na določeno elektrodo se lahko ob različnih časovnih intervalih prestavijo različne kapljice, zato moramo pomniti položaje vseh kapljic ob različnih časih.

Pri potovanju po elektrodnem polju poskušajo nekateri algoritmi zmanjšati število uporabljenih elektrod, saj to poveča robustnost in preprečuje, da bi potovale

kapljice po elektrodah, ki ne delujejo pravilno. Problem, ki nastane pri transportu kapljice po elektrodnem polju, je puščanje sledi po poti, kjer je potovala kapljica. Nekateri algoritmi to upoštevajo že pri samem usmerjanju in se poskušajo izogniti prečkanju sledi dveh kapljic med seboj. Drugi algoritmi v ta namen uporabljajo čistilne kapljice, če to DMFB omogoča, ki jih algoritem usmeri tako, da sčistijo sledi prejšnjih kapljic in nato odpadke zavržejo v odpadni rezervoar.

3.4.2 Omejitve pri usmerjanju

Kapljica je v 3D prostoru predstavljena z 2D položajem na elektrodnem polju in časom. Na primer kapljica d_i ima koordinate (X_i, Y_i, t), kjer X_i in Y_i predstavljata koordinati kapljice na elektrodnem polju, indeks i predstavlja številko kapljice, t pa diskreten čas. 3D prostor je primeren predvsem zaradi potrebe po zveznosti poti vsake kapljice skozi čas. Pri usmerjanju kapljic je treba upoštevati tudi dve omejitvi [13, 12].

Prva je *omejitev tekočine*, ki preprečuje, da bi dve kapljici prišli na dve sosednji ali eno isto elektrodo, razen v primeru namernega mešanja kapljic. Položaj kapljice je v naslednjem časovnem koraku lahko na petih različnih položajih:

$$(X_i, Y_i, t+1)$$
$$(X_i+1, Y_i, t+1)$$
$$(X_i-1, Y_i, t+1)$$
$$(X_i, Y_i+1, t+1)$$
$$(X_i, Y_i-1, t+1)$$

Zato se v 3D predstavitvi okoli nje naredi navidezna kocka, ki preprečuje drugim kapljicam, da bi prišle v njeno bližino. Zapis položaja kapljice bomo zaradi lažjega zapisa in razumevanja spremenili v 2D koordinate, ki sta odvisni od časa. Tako ima kapljica d_i s položajem (X_i, Y_i, t) sedaj koordinate $(X_i(t), Y_i(t))$ in kapljica d_j koordinate $(X_j(t), Y_j(t))$.

Zaradi vzporednih premikov kapljic morajo za poljubni dve kapljici d_i in d_j veljati naslednje omejitve:

- statična omejitev:

$$\begin{aligned} |X_i(t+1) - X_j(t+1)| &\geq 2 \\ \vee\ |Y_i(t+1) - Y_j(t+1)| &\geq 2 \end{aligned} \tag{3.3}$$

- dinamični omejitvi:

$$\begin{aligned} |X_i(t+1) - X_j(t)| &\geq 2 \\ \vee\ |Y_i(t+1) - Y_j(t)| &\geq 2 \end{aligned} \tag{3.4}$$

$$\begin{aligned} |X_i(t) - X_j(t+1)| &\geq 2 \\ \vee\ |Y_i(t) - Y_j(t+1)| &\geq 2 \end{aligned} \tag{3.5}$$

Te omejitve preprečujejo, da bi se kapljici približali tako po horizontalnih, vertikalnih in diagonalnih položajih [12]. Statična omejitev 3.3 zagotavlja, da je absolutna razdalja med dvema kapljicama tako po X in Y osi vedno večja ali enaka 2. Prva dinamična omejitev 3.4 skrbi, da se kapljica d_i v naslednjem časovnem koraku ne približa kapljici d_j v trenutnem časovnem koraku po X in Y osi za manj kot 2 mesti. Druga dinamična omejitev 3.5 omejuje razdaljo med kapljico d_i v trenutnem časovnem koraku in kapljico d_j v naslednjem časovnem koraku po X in Y osi na najmanj 2 mesti. Za premik kapljice morajo biti izpolnjene vse tri omejitve (3.3, 3.4 in 3.5), saj lahko sicer pride do neskladja z zahtevami za izvedbo preizkusa. Modulom, ki so postavljeni na elektrodno polje, moramo dodati elektrode okoli njega, ki preprečujejo mešanje in vdiranje kapljic v sam modul.

Druga je *časovna omejitev*, saj je za vsako kapljico določeno, kdaj mora biti na svojem cilju. Vendar ta omejitev ne povzroča toliko težav, saj se kapljica giblje po elektrodnem polju zelo hitro, tudi do 20 cm/s, zato pri nekaterih izračunih časa potovanja kapljice ne upoštevamo v končnem izračunu.

3.4.3 Splošni algoritem

Glavna cilja *algoritma za usmerjanje kapljic* [13] sta usmerjanje in transport kapljic v predvidenem času. Glavni ideji, da dosežemo omenjena cilja, sta *obvoz ovir* in *umikanje kapljic*. Obvoz ovir temelji na analizi števila kapljic, ki bodo enostavno prišle mimo vseh ovir in kapljic, ki so že bile usmerjene. Z idejo obvoza ovir

usmerimo vse kapljice, ki lahko pridejo brez težav do ciljev, ostanejo pa nam kapljice, ki ne morejo do svojih ciljev zaradi različnih ovir. Za te kapljice poskrbi funkcija umikanja kapljic. Funkcija namreč umakne kapljice, ki povzročajo zastoj, in omogoči prehod.

Ideja algoritma je podobna cestnemu prometu, kjer veljajo določena pravila in zahteve. Na primer, določena je razdalja med dvema voziloma, pri srečanju dveh nasproti si vozečih vozil se mora eden pred ožino umakniti, da lahko prideta skozi ožino. Ustavljeno vozilo na prometnem cestišču bo povzročilo zastoj.

Tako se tudi to upošteva v predstavljenem algoritmu in naenkrat usmerjamo samo eno kapljico. Na koncu se vse poti sestavijo skupaj.

Algoritem 2: Splošni algoritem

```
1  Sekvenčni graf G = (V, E)
2  Nabor vseh kapljic D_u = D
3  Časovna omejitv RT
4  T_b = 0
5  T_c = 0
6  repeat
7  |  T_b = Routing-Bypassibility(D_u, G, max(T_b, T_c))
8  |  if T_b se ne poveča then
9  |  |  T_c = max(Routing-Concession(D_u, G, T_b), T_c)
10 |  end if
11 until Ne usmerimo vseh kapljic
12 Routing-Compaction(D_u, D, G, RT)
```

Splošni algoritem 2 ima za vhodne podatke sekvenčni graf G, nabor vseh kapljic D in časovno omejitev za usmerjanje kapljic RT. V vrstici 4 nastavimo osnovni čas usmerjanja T_b na 0. Čas usmerjanja kapljice z umikanjem T_c nastavimo na 0 v vrstici 5. Nato usmerjamo kapljico z obvozom ovir v vrstici 7 do njenega cilja. Uspešna usmeritev kapljice zmanjšuje število kapljic, ki jih moramo usmeriti.

V primeru, ko se osnovni čas usmerjanja T_b ne poveča, pomeni, da je prišlo do zastoja oziroma ovire, ki je ne moremo preprosto rešiti. Takrat uporabimo funkcijo za umikanje kapljic v vrstici 9, ki sprosti pot. Nato nadaljujemo z usmerjanjem kapljic z največjo zmožnostjo obvoza ovir. Na koncu se v vrstici 12 izvede algoritem, ki poskuša vse poti ponovno usmeriti.

3.4.4 Obvoz ovir

Kapljica d_i, ki prispe na svoj cilj (angl. target) T_i, tam tudi ostane. Okoli nje nastane obroč elektrod, ki se ne smejo aktivirati. To lahko na gostem omrežju povzroči daljšo pot ostalim kapljicam ali jim zapre pot. Zato je bolje, da take kapljice usmerimo kasneje in omogočimo prehod drugim. Koncept obvoza ovir predlaga zajemanje zastojev okoli ciljnih položajev kapljic in tako preverja, ali bodo lahko druge kapljice prišle mimo. Možni so štirje obvozi mimo ovire H_{zgoraj}, H_{spodaj}, V_{levo} in V_{desno}, kjer H pomeni horizontalno in V vertikalno. Izjema, ki ne povzroča zastojev in ga ni potrebno obvoziti, je odpadni rezervoar, kjer se shranjujejo ena ali več odpadnih kapljic.

Glede na blokiranost oziroma neblokiranost poti kapljice, lahko obvoze ovir razdelimo v štiri kategorije:

Idealna zmožnost obvoza je samo takrat, ko je cilj kapljice odpadni rezervoar. Kapljice v odpadnem rezervoarju ne povzročajo novih ovir za usmerjanje še neusmerjenih kapljic.

Polna zmožnost obvoza omogoča kapljici vsaj en vertikalen in en horizontalen obvoz ovire. Kapljica ima vsaj dve možnosti za obvoz ovire, ki ji omogočata prihod na cilj.

Polovična zmožnost obvoza omogoča kapljici vsaj en vertikalen ali en horizontalen obvoz ovire. Kapljica bo lahko prišla mimo ovire do svojega cilja.

Ni zmožnosti obvoza ne omogoča kapljici nobenega vertikalnega ali horizontalnega obvoza ovire. To pomeni, da ovira blokira pot kapljici in s blokado povzroča zastoj.

Ob usmeritvi kapljice z idealno ali polno zmožnostjo obvoza, to ne bo imelo posledic na zmožnost obvoza, ker si lahko kapljice delijo obvoze glede na različne čase, kar pripelje do dveh izrekov.

Izrek 3.1 *Usmerjanje kapljice z idealno zmožnostjo obvoza ne more vplivati na splošno zmožnost usmerjanja na čipu ali povečati Manhattanske*[3] *dolžine poti še neusmerjene kapljice v 2D ravnini.*

[3]Manhattanska dolžina (angl. manhattan length) je razdalja med dvema točkama v 2D ravnini, kjer so možni samo vertikalni in horizontalni premiki.

Dokaz. Imamo dve kapljici d_i in d_j, ki še nista bili usmerjeni in naj imata obe dopustni poti $P_i(t)$ in $P_j(t)$ ob času t. Naj ima kapljica d_i idealno zmožnost obvoza in tako ne ustvari novih zastojev. Tako ima d_j še zmeraj prosto pot $P_j(AT_i+1)$ ob času AT_i+1, kjer je AT_i čas prihoda kapljice d_i na cilj. Zato je lahko Manhattanska dolžina poti $P_j(AT_i+1)$ enako dolga kot tista od $P_j(t)$ v 2D ravnini. □

Izrek 3.2 *Usmerjanje kapljice s polno zmožnostjo usmerjanja ne vpliva na zmožnost obvoza na elektrodnem polju, lahko pa poveča Manhattansko dolžino poti neusmerjene kapljice od izvora do cilja v 2D ravnini.*

Dokaz. Imamo dve kapljici d_i in d_j, ki še nista bili usmerjeni in naj imata obe dopustni poti $P_i(t)$ in $P_j(t)$ ob času t. Kapljica d_i naj ima polno zmožnost obvoza in zato kapljica d_i na svojem cilju T_i postavi blokado B ob času AT_i-1. Vendar ima d_j še zmeraj dopustno pot $P_j(AT_i+1)$ ob času AT_i+1. Zato je lahko Manhattanska dolžina poti $P_j(AT_i+1)$ daljša ali enako dolga kot od $P_j(t)$, zaradi blokade B v 2D ravnini. □

Algoritem 3: Routing-Bypassibility

```
1  Sekvenčni graf G = (V, E)
2  Nabor neusmerjenih kapljic D_u
3  Osnovni čas usmerjanja T_b
4  S = razvrstimo D_u v padajočem vrstnem redu glede na zmožnost obvoza
   ovir
5  foreach d_i ∈ S do
6      P = 2D minimalna dolžina poti za d_i po času T_b
7      if P ≠ ∅ then
8          Usmeri d_i s P
9          D_u = D_u \ {d_i}
10         return AT_i + 1
11     end if
12 end foreach
13 return T_b
```

Algoritem 3 ima za vhodne podatke podan sekvenčni graf G, nabor še neusmerjenih kapljic D_u in osnovni čas usmerjanja T_b. V vrstici 4 najprej razvrstimo neusmerjene kapljice glede na zmožnost obvoza ovir v padajočem vrstnem redu,

kjer ima kapljica z idealno zmožnostjo obvoza ovir najvišjo vrednost, kapljica, ki nima zmožnosti obvoza ovir, pa najnižjo vrednost. Razvrščene neusmerjene kapljice shranimo v S. Nato v zanki izberemo vsako kapljico d_i iz seznama S in ji v vrstici 6 poiščemo minimalno dolžino poti po osnovnem času T_b.

V primeru, da je pot P različna od $\emptyset$, kapljico d_i usmerimo po poti P. Usmerjeno kapljico moramo sedaj odstraniti s seznama neusmerjenih kapljic D_u, v vrstici 9. Nato posodobimo osnovni čas usmerjanja T_b (glej Algoritem 2) tako, da v vrstici 10 vrnemo končni čas usmerjanja $AT_i + 1$.

Če je pot P enaka $\emptyset$, izberemo naslednjo kapljico in ji poskušamo poiskati pot. Ko kapljico d_i pripeljemo na cilj T_i, morajo ciljna elektroda in elektrode okoli cilja ostati neaktivne, dokler kapljice ponovno ne usmerimo. S določanjem neaktivnih elektrod preprečimo nenadzorovano mešanje kapljic. V primeru, da nobene kapljice ne uspemo usmeriti, vrnemo v vrstici 13 enak osnovni čas T_b, kot smo ga prejeli.

3.4.5 Umikanje kapljic

Zaradi kompleksnosti DMFB-ja in naivnega zaporednega usmerjanja kapljic prihaja do napak, kot je zastoj. Zastoj nastane, ker ena kapljica zapira pot drugi, da bi prišla do svojega cilja. Tega ni možno rešiti ne v 2D ravnini ali 3D prostoru. Ta problem bi lahko rešili tako, da bi dve kapljici premikali hkrati, kar pripelje do vzporednega usmerjanja in velike računske zahtevnosti. Obstaja zaporedna rešitev, opisana v algoritmu 4, v kateri se kapljica, ki ovira prehod, prestavi na proste elektrode. Te proste elektrode imenujemo *območje popuščanja* (angl. cocession zone), kjer kapljica počaka, dokler ne gredo ostale kapljice mimo. Območje popuščanja je skupek vsaj treh prostih elektrod skupaj.

Algoritem 4 je podoben algoritmu 3. Razlikujeta se v vrstici 4, kjer v algoritmu 4 razvrščamo D_u glede na razdaljo do najbližjega območja popuščanja. Druga razlika je v vrstici 6, kjer v algoritmu 4 pot P izračunamo kot minimalno dolžino 3D poti po osnovnem času T_b, kateremu prištejemo dodaten čas α_i:

$$\alpha_i = \sum_{j \in B_i \cap D_u} \mid X_{j,izvor} - X_{j,cilj} \mid + \mid Y_{j,izvor} - Y_{j,cilj} \mid,$$

kjer je B_i seznam kapljic, ki jim kapljica d_i ovira pot. 2D koordinate ($X_{j,izvor}$, $Y_{j,izvor}$) predstavljajo položaj izvora kapljice d_j in $(X_{j,cilj}, Y_{j,cilj})$ položaj cilja ka-

pljice d_j. Čas α_i tako predstavlja vsoto Manhattanskih razdalj med koordinatami izvora in cilja kapljice j, kjer j pripada preseku med B_i in D_u.

V primeru, da je pot P različna od $\emptyset$, kapljico usmerimo in ji umaknemo kapljico, ki ovira njeno pot na območje popuščanja, ali pa poskušamo usmeriti drugo kapljico. Kapljice, ki smo jih umaknili, morajo počakati na območju popuščanja, dokler kapljica, ki jo usmerjamo, ne doseže svojega cilja. Nato kapljice, ki smo jih umaknili, premaknemo nazaj na njihovo prejšnje mesto.

Algoritem 4: Routing-Concession

```
1  Sekvenčni graf G = (V, E)
2  Nabor neusmerjenih kapljic D_u
3  Osnovni čas usmerjanja T_b
4  S = razvrstimo D_u v padajočem vrstnem redu glede na razdaljo do
   najbližjega območja popuščanja
5  foreach d_i ∈ S do
6  |  P = 3D minimalna dolžina poti za d_i po času T_b + α_i
7  |  if P ≠ ∅ then
8  |  |  Usmeri d_i s P
9  |  |  D_u = D_u \ {d_i}
10 |  |  return AT_i + 1
11 |  end if
12 end foreach
13 return T_b
```

3.4.6 Zgoščevanje usmeritve

V algoritmih 3 in 4 vedno usmerjamo samo eno kapljico v določenem časovnem intervalu, kar lahko pripelje do kršitve časovnih omejitev. Zato poskušamo v algoritmu 5 preusmeriti vse kapljice s požrešnim načinom in tako povečati izrabo modulov ter zadovoljiti časovne omejitve, ne zmanjšamo pa zmožnosti usmerjanja. S preusmerjanjem kapljic pripomoremo k uporabi manjšega števila elektrod, to pa zmanjša možnosti za uporabo nedelujočih elektrod.

V algoritmu 5, v vrstici 6, vsem še neusmerjenim kapljicam nastavimo čas prihoda AT na neskončno. Nato ponavljamo vrstice med 8 in 23, dokler ne zaznamo,

da ni napredka ali dosežemo maksimalno število iteracij. Vse kapljice v vrstici 9 razvrstimo glede na AT v padajočem vrstnem redu. Zanka med vrsticama 10 in 22 je sestavljena iz dveh delov.

V prvem delu, med vrsticama 11 in 16, skrbi za zadovoljevanje časovne omejitve usmerjanja kapljic. V vrstici 11 preverimo ali je maksimalni čas prihoda katerekoli kapljice večji od časovne omejitve. Če je večji, v vrstici 12 določimo kapljici d_i minimalno dolžino poti P. V primeru, da pot P obstaja in se čas prihoda kapljice d_i zmanjša, kapljico usmerimo po poti P.

Algoritem 5: Routing-Compaction

```
1  Sekvenčni graf G = (V, E)
2  Nabor neusmerjenih kapljic D_u
3  Nabor vseh kapljic D
4  Časovna omejitv RT
5  foreach d_i ∈ D_u do
6  |  AT_i = ∞
7  end foreach
8  repeat
9  |  S = razvrstimo D v padajočem vrstnem redu glede na AT
10 |  foreach d_i ∈ S do
11 |  |  if RT < max{AT_i|∀i} then
12 |  |  |  P = 3D minimalna dolžina poti za d_i glede na časovno omejitev
13 |  |  |  if P ≠ ∅ ∧ AT_i se bo zmanjšal then
14 |  |  |  |  Usmeri d_i s P
15 |  |  |  end if
16 |  |  else
17 |  |  |  P = 3D minimalna dolžina poti za d_i glede na toleranco napak
18 |  |  |  if AT_i ≤ RT then
19 |  |  |  |  Usmeri d_i s P
20 |  |  |  end if
21 |  |  end if
22 |  end foreach
23 until Ni napredka ali maksimalno število iteracij
```

Če je časovna omejitev večja od maksimalnega časa prihoda katerekoli kapljice na cilj, se izvede drugi del, ki povečuje toleranco napak. Tudi v drugem delu moramo, v vrstici 17, izračunati 3D minimalno dolžino poti P. Vendar mora za usmeritev d_i s potjo P v drugem delu veljati tudi pogoj v vrstici 18, kjer se preveri ali je končni čas prihoda kapljice d_i na cilj manjši od časovne omejitve. V primeru da je, kapljico usmerimo po poti P. *Toleranca napak* je zmožnost računalniškega sistema, da po kakršnikoli napaki strojne ali programske opreme deluje še naprej brez človekovega posega. Delovati mora na dani ravni, ki zagotavlja nadaljevanje delovanja in integriteto podatkov.

3.5 Simulator

Programska oprema, ki jo uporabljamo za *statično simulacijo sinteze* (SSS) je bila implementirana na Univerzi v Kaliforniji, Riverside (UCR). Namenjena je simulaciji metod, ki se uporabljajo za sintezo na DMFB-ju, in vizualizaciji rezultatov. Omogoča tudi razvijalcem prijazno in uporabno razvojno okolje za razvijanje novih algoritmov za razvrščanje operacij, postavljanje modulov in usmerjanje kapljic.

Novo razvite algoritme lahko primerjamo tudi z ostalimi že implementiranimi. Simulator je razdeljen na dva nivoja. Prvi je nivo sinteze, napisan v jeziku C++, zaradi hitrejšega izvajanja algoritmov in s tem tudi preizkusov. Drugi je vizualizacijski nivo, napisan v jeziku Java, ki nam omogoča vizualizacijo podatkov in izbiro različnih preizkusov.

3.5.1 Priprava razvojnega okolja in namestitev

Za delovanje DMFB SSS in razvijanje novih algoritmov si moramo najprej namestiti nekaj programov in knjižnic. Opisan postopek je namenjen operacijskemu sistemu Windows. S spletne strani http://www.microfluidics.cs.ucr.edu prenesemo zadnjo verzijo izvorne kode DMFB SSS.

MinGW Za izvajanje sinteze, napisane v C++ jeziku, moramo prenesti in namestiti zadnjo verzijo programa MinGW [4] in namestiti C++ prevajalnik. Po namestitvi moramo nastaviti sistemske spremenljivke okolja:

1. odpremo “Sistemske lastnosti”, kjer izberemo zavihek “Dodatno”,

2. kliknemo na gumb “Spremenljivke okolja...”,
3. odpre se nam novo okno “Spremenljivke okolja”, kjer med sistemskimi spremenljivkami poiščemo spremenljivko “Path”, jo izberemo in kliknemo “Uredi ...”,
4. spremenljivki “Path” v polje “Vrednost spremenljivke” na konec dodamo “`;C:\MinGW\bin`” brez narekovajev,
5. nato ponovno zaženemo računalnik.

Eclipse Sledi namestitev okolja “Eclipse”. Zadnjo različico programa “Eclipse Standard” prenesemo na računalnik in ga namestimo. Namestiti moramo še “Eclipse CDT”, ki podpira razvojno okolje za C++:

1. s spletne strani [1] za “Eclipse CDT” kopiramo naslov za “p2 software repository” za našo različico “Eclipse-a”,
2. v programu “Eclipse” odpremo “Help → Install New Software...”, prilepimo naslov v polje “Work with:” in pritisnemo “Enter” na tipkovnici,
3. v spodnjem oknu izberemo “CDT Main Features” in pritisnemo “Next”,
4. izberemo vse možnosti in pritisnemo “Next”,
5. izberemo ”I accept the terms of the license agreement”in kliknemo ”Finish”,
6. pojavi se okno za ponovni zagon programa, kjer kliknemo ”Yes”.

Java Developer Kit Za delovanje izvorne kode napisane v jeziku ”Java”, moramo prenesti in namestiti zadnjo različico “Java Developer Kit (JDK)” [2]. Za vizualizacijo sinteze moramo dodati “Java 3D” knjižnico v naš sistem. Glede na to, katero različico JDK (32/64bit) smo namestili, moramo prilagoditi tudi “Java 3D”. Kopirati moramo “Java 3D” “`dlls`” in “`jar`” datoteke v primerne podmape “`\bin`” in “`\lib\ext`” v mapi JRE:

- [64bit JRE] kopiramo iz “`MFSimStatic_Source\Shared\Java3D\64`” v JRE mapo (verjetno “`C:\Program Files\Java\jreX`”),
- [32bit JRE] kopiramo iz “`MFSimStatic_Source\Shared\Java3D\32`” v JRE mapo (verjetno “`C:\Program Files (x86)\Java\jreX`”).

Uvoz in nastavitev projektov v Eclipse Ko smo pripravili okolje za DMFB SSS, moramo nastaviti še projekte. V programu "Eclipse" kliknemo na "File → Switch Workspace → Other..." in izberemo mapo, ki vsebuje "Java" in "C++" projekti ("`MFSimStatic_Source/`"). Nato kreiramo "Java" projekt za vizualizacijo sinteze:

1. kliknemo na "File → New → Java Project",
2. projektu v "Project name" dodamo ime "DmfbSimVisualizer", kar bo avtomatsko uvozilo obstoječe datoteke,
3. kliknemo "Finish".

Drugi "Java" projekt, ki ga moramo kreirati, je grafični vmesnik za izvajanje sinteze:

1. kliknemo na "File → New → Java Project",
2. projektu v "Project name" dodamo ime "MFSimStaticGUI", kar bo avtomatsko uvozilo obstoječe datoteke,
3. kliknemo "Finish".

Zadnji je C++ projekt, kjer se izvajajo algoritmi in celotna sinteza:

1. kliknemo na "File → New → Project...",
2. razširimo mapo "C/C++", izberemo "C++ Project" in pritisnemo "Next",
3. projektu v "Project name" dodamo ime "MFSimStatic", kar bo avtomatsko uvozilo obstoječe datoteke,
4. za "Project type" izberemo "Executable → Empty Project",
5. za "Toolchains" izberemo "MinGW GCC",
6. kliknemo "Finish",
7. ko vpraša ali odpre C++ perspektivo, izberemo "Yes".

Dodati moramo še referenco za časovno knjižnico:

1. z miško desno kliknemo na “MFSimStatic” C++ projekt v “Project Explorer”,
2. na seznamu, ki se nam prikaže izberemo “Properties”,
3. odpre se nam novo okno “Properties for MFSimStatic”, kjer kliknemo “C/C++ Build → Settings → MinGW C++ Linker → Libraries”,
4. v izbirnem polju “Configuration” izberemo “[All configurations]”,
5. v “Libraries(-l)” dodamo “`winmm`” in pritisnemo “Apply” in še “OK”,
6. to bo odstranilo nedefinirane reference napak.

Za zagon prve simulacije moramo najprej prevesti preprost primer, tako da v programu “Eclipse” z uporabo “Project Explorer” odpremo “MFSimStatic → Source → main.cc” v perspektivi za “C/C++”. Kliknemo na kladivo v orodni vrstici, da prevedemo razvijalsko različico simulatorja. Nato moramo kreirati zagonsko konfiguracijo:

1. kliknemo na “Run → Run Configuration...”,
2. odpre se nam novo okno “Run Configurations”,
3. v levem delu okna dvakrat kliknemo na “C/C++ Application”, da kreiramo novo konfiguracijo,
4. v polje “Name” vpišemo “MFSimStatic Debug” in preverimo da je “`Debug\MFSimStatic.exe`” vpisano v polje “C/C++ Application”,
5. nato kliknemo “Close”.

Simulacijo lahko sedaj zaženemo z klikom na ”Run → Run As → Local C/C++ Application”.

3.5.2 Uporaba simulatorja

Prva različica simulatorja je bila predstavljena novembra leta 2012. Vsebovala je samo algoritem za razvrščanje seznama (angl. List scheduling), privzet algoritem za postavljanje modulov, v katerem je bilo mesto modula na elektrodnem polju že

prej definirano, in Dijkstra algoritem za usmerjanje vsake kapljice posebej. Sledil je razvoj različnih algoritmov, ki so optimizirali porabljen čas.

Štirim dodiplomskim in enemu podiplomskemu študentu, ki je bil tudi glavni oblikovalec, je bil predstavljen simulator. Študenti so dobili dokumentacijo za različne algoritme in splošne informacije o DMFB. Inštruktor jim je pomagal, da so razumeli vse psevdokode za algoritme, ki so jih nato implementirali. Problem razvijanja novih algoritmov je v tem, da je vsak posebej razvit v izoliranem okolju in ne skupaj z ostalimi, saj so med seboj odvisni.

Ta simulator podpira ravno to, da so algoritmi lahko razviti ločeno, saj je definiran prehod med algoritmi. Težko napovemo, ali bodo algoritmi skupaj podali uporabno rešitev, v kateri ne bo prišlo do napak med samim izvajanjem.

Simulator je modularen, saj se vsak algoritem lahko izvede posamično ali se izvedejo vse stopnje naenkrat. Moramo pa izvesti vse stopnje pred algoritmom, ki ga želimo izvesti. Če želimo, na primer, izvesti algoritem postavljanja modulov, moramo najprej izvesti razvrščanje operacij.

Simulator ima natančno opredeljene podatkovne strukture znotraj vsakega algoritma in med vsako stopnjo. Med stopnjami simulator izdela tekstovno datoteko z vsemi podatki za vizualizacijo oziroma izvedbo naslednje stopnje sinteze. To omogoča hitro odpravljanje napak in primerjave med različnimi algoritmi in njihovimi rezultati.

Različica, ki jo uporabljamo v diplomski nalogi, je v3.1 in je bila izdana 2. julija 2014. Ta različica vsebuje različne že implementirane algoritme za sintezo. Za razvrščanje operacij so implementirani algoritmi, kot so:

- List scheduler,
- Genetic algorithm scheduler,
- Rickett's genetic algorithm scheduler,
- O'Neal's force directed list scheduler.

Implementiran je tudi algoritem za razvrščanje poti, ki je opisan v poglavju 3.2. Algoritma za postavljanje modulov, ki sta implementirana:

- KAMER linked-list placement,

- Calvin & Skyler's routing based synthesis placer.

Algoritem simuliranega ohlajanja, ki sem ga opisal v poglavju 3.3, je tudi implementiran. Za usmerjanje kapljic pa so implementirani sledeči algoritmi:

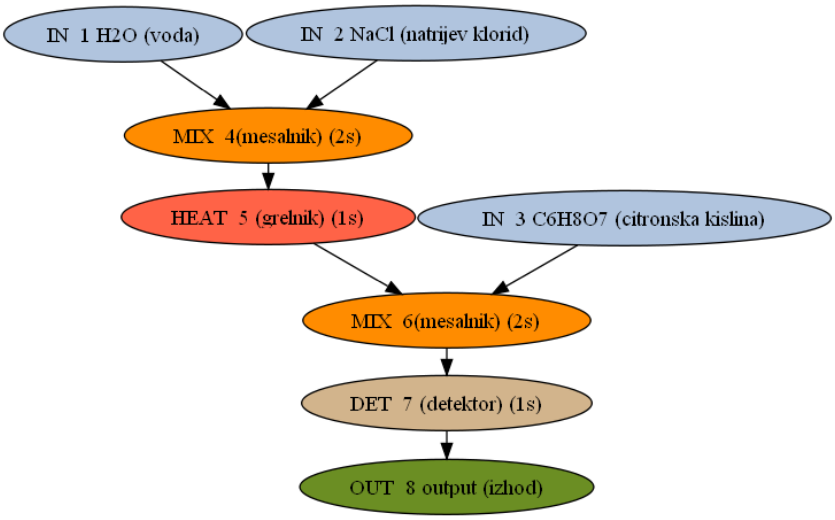
- A* router,
- Lee's router,
- Grissom's simplified roy maze router,
- Yuh's BioRouter,
- Grissom's fixed-palce router,
- Grissom's fixed-place map router,
- Calvin & Skyler's routing based synthesis router.

Chojev algoritem za usmerjanje kapljic, predstavljen v poglavju 3.4, je prav tako implementiran, vendar z razliko, da uporablja za računanje individualnih poti algoritem Dijkstra in ne Manhattanske razdalje, kot je opisano v diplomskem delu.

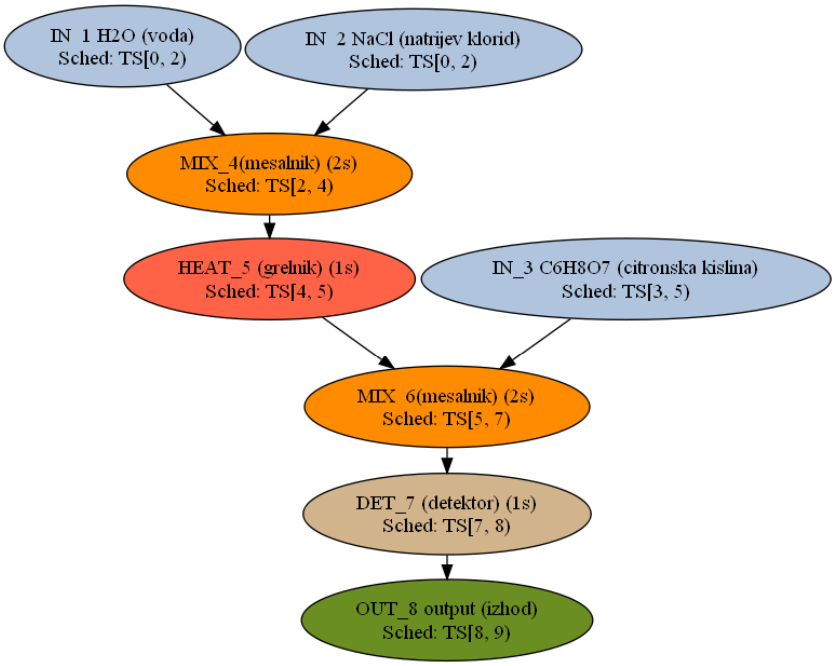
3.5.3 Primer simulacije

Simulator, opisan v prejšnjih dveh podrazdelkih, smo uporabili za izvedbo preprostega primera preizkusa. Preizkus ima za delovanje na voljo sledeče module: tri vhodne rezervoarje, en mešalnik velikosti 2×4 elektrod, en grelnik in en detektor v velikosti 4×3 elektrod ter en izhodni rezervoar. Določena je velikost elektrodnega polja, 11×11 elektrod, in najdaljši čas izvajanja 10 sekund.

Poleg tega je podan tudi sekvenčni graf operacij in odvisnosti med njimi, skupaj z časom izvajanja vsake operacije, kot to prikazuje Slika 3.3. Vozlišča modre barve v sekvenčnem grafu predstavljajo vhodne rezervoarje s podano snovjo, ki je v rezervoarju. Oranžna vozlišča predstavljajo mešalne operacije s potrebnim časom izvajanja operacije. Z rdečo barvo je označeno vozlišče, ki predstavlja grelno operacijo in potreben čas gretja. Predzadnje vozlišče oker barve predstavlja detektor in potreben čas detekcije v oklepaju. Zadnje vozlišče zelene barve pa predstavlja izhodni rezervoar.

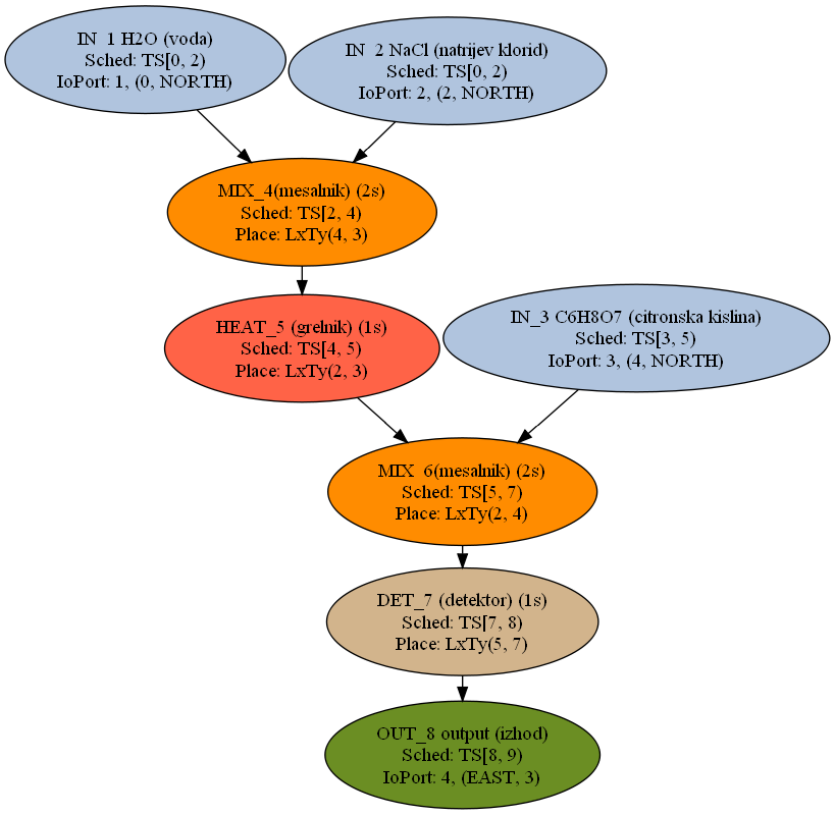


Slika 3.3: Nerazvrščen sekvenčni graf.



Slika 3.4: Razvrščen sekvenčni graf.

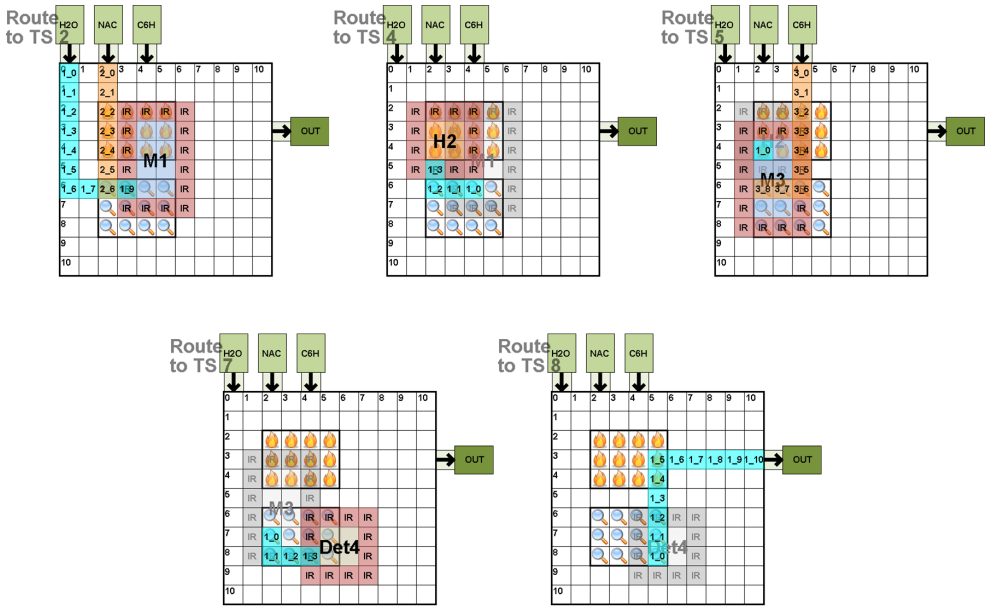
Z algoritmom razvrščanja poti smo razvrstili operacije v sekvenčnem grafu in jim določili začetne ter končne čase izvajanja, kar prikazuje Slika 3.4. Vsakemu vozlišču se doda oznaka TS, ki predstavlja časovni korak (angl. time-step) v katerem se operacija začne in konča.



Slika 3.5: Razvrščen sekvenčni graf s položaji modulov na elektrodnem polju.

Sledi algoritem za postavljanje modulov na osnovi simuliranega ohlajanja. Modulom se določi njihov položaj za časovne korake predpisane v razvrščenem sekvenčnem grafu na Sliki 3.4. Vozliščem, ki predstavljajo vhodne oziroma izhodne rezervoarje, dodamo zaporedno številko vhodno-izhodnih vrat (angl. port). V oklepaje dodamo položaj rezervoarja glede na regije (sever, jug, vzhod oziroma zahod) ter številko, ki predstavlja X oziroma Y koordinato elektrode ob robu elektrodnega polja. Koordinatno izhodišče je postavljeno v levi zgornji kot elektrodnega polja. Na abscisni osi je predstavljena koordinata X, na ordinatni osi pa Y. Ostalim vozliščem, to so operacije mešanja, gretja in detekcije, dodamo oznako

$LxTy$, kjer Lx predstavlja najbolj levo X koordinato in Ty najvišjo Y koordinato modula. Za oznako $LxTy$ sta v oklepaju napisani številki Lx in Ty, kot prikazuje Slika 3.5.



Slika 3.6: Poti kapljic v različnih časovnih korakih.

Zadnji algoritem usmeri vse kapljice od njihovega izvora do cilja. Na Sliki 3.6 so z modro in oranžno barvo označene poti kapljic. Po vsaki označeni poti, v časovnem koraku TS, potuje ena kapljica od svojega izvora do cilja. Vsaka elektroda, po kateri potuje kapljica, je označena s številko kapljice in številom premikov, ki jih je na svoji poti naredila kapljica.

Čas izvajanja vsakega algoritma se zapiše v datoteko za spremljanje izvajanja simulacije. Za primer preizkusa, ki smo ga izvedli, se je razvrščanje operacij izvajalo 0 ms, postavljanje modulov 2 ms in usmerjanje kapljic 0 ms.

Poglavje 4

Zaključek

V diplomskem delu so predstavljene osnove mikrofluidike in laboratorijev na čipu, ki sta osnovi za razvoj digitalnih mikrofluidnih biočipov. Digitalni mikrofluidni biočipi so razdeljeni na strojni in programski nivo. Predstavljen je tudi celostni pogled na sintezo, s katero omogočamo delovanje digitalnega mikrofluidnega biočipa. Predstavljeni so algoritmi za razvrščanje poti, ki rešuje problem razvrščanja operacij, simulirano ohlajanje za postavljanje modulov na elektrodno polje in algoritem za usmerjanje kapljic.

Na koncu je predstavitev simulatorja, ki simulira delovanje digitalnega mikrofluidnega biočipa in v katerem so implementirani predstavljeni algoritmi. Z simulatorjem smo izvedli tudi preprost primer preizkusa. V sintezi preizkusa smo uporabili algoritme, ki smo jih opisali v diplomskem delu.

Področje digitalnih mikrofluidnih biočipov se bo v naslednjih letih še naprej razvijalo, saj je potreba po izvajanju različnih preizkusov iz dneva v dan večja. Napredovala bo tudi v smeri manjšanja velikosti čipov in porabe energije, ki je glavna težava pri prenosljivih napravah. Seveda se bodo razvijali tudi novi algoritmi in metode za potrebe izvajanja sinteze.

Literatura

[1] Eclipse CDT. http://www.eclipse.org/cdt/downloads.php, 2014.

[2] Java Developer Kit. http://www.oracle.com/technetwork/java/javase/downloads/index.html, 2014.

[3] Laboratorij na čipu. http://www.genome.gov/dmd/img.cfm?node=Photos/Technology/Genome%20analysis%20technology&id=79284, 2014.

[4] MinGW. http://sourceforge.net/projects/mingw/files/latest/download?source=files, 2014.

[5] Zvezni pretočni mikrofluidni biočip. http://en.wikipedia.org/wiki/Microfluidics#mediaviewer/File:Microfluidics.jpg, 2014.

[6] Bhargab B Bhattacharya, Sudip Roy in Sukanta Bhattacharjee. Algorithmic Challenges in Digital Microfluidic Biochips: Protocols, Design, and Test. In *Applied Algorithms*. Springer, 2014.

[7] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[8] Daniel Grissom in Philip Brisk. Path scheduling on digital microfluidic biochips. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012.

[9] Daniel T Grissom in Philip Brisk. Fast Online Synthesis of Digital Microfluidic Biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(3):356–369, 2014.

[10] Fei Su in Krishnendu Chakrabarty. Module placement for fault-tolerant microfluidics-based biochips. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 11(3):682–710, 2006.

[11] Fei Su, Krishnendu Chakrabarty in Richard B Fair. Microfluidics-based biochips: technology issues, implementation platforms, and design-automation challenges. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(2):211–223, 2006.

[12] Fei Su, William Hwang in Krishnendu Chakrabarty. Droplet routing in the synthesis of digital microfluidic biochips. In *Proceedings Design Automation and Test in Europe*. IEEE, 2006.

[13] Minsik Cho in David Z Pan. A high-performance droplet routing algorithm for digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1714–1724, 2008.

[14] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi in drugi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[15] Tsung-Yi Ho, Jun Zeng in Krishnendu Chakrabarty. Digital microfluidic biochips: A vision for functional diversity and more than Moore. In *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2010.

[16] Shailendra Yadav. Analysis of value creation and value capture in microfluidics market. *Doktorska disertacija*, 2010.